

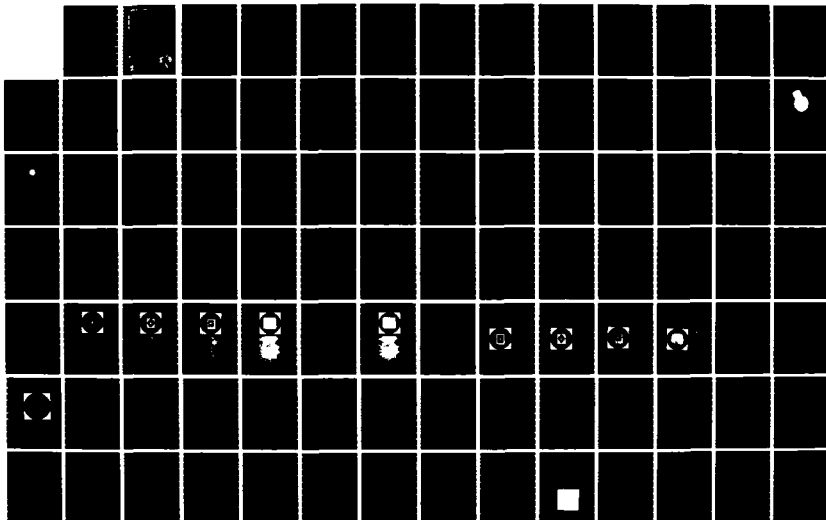
AD-A168 521

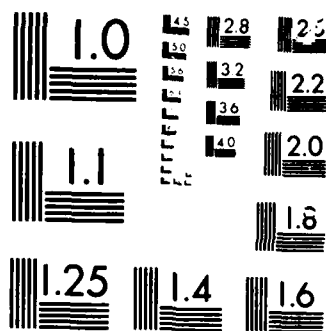
BIOLOGICAL VISUAL SYSTEMS STRUCTURES FOR MACHINE VISION 1/4
APPLIED TO ROBOTI.. (U) VIRGINIA UNIV CHARLOTTESVILLE
DEPT OF ELECTRICAL ENGINEERING. R M INIGO ET AL.

UNCLASSIFIED

FEB 86 UVA/525647/EE86/101 AFOSR-TR-86-0202 F/G 6/4

NL





Final Report
AFOSR-84-0000
**NEURAL VISUAL SYSTEM STRUCTURES
FOR
MACHINE VISION APPLIED TO ROBOTICS**

Submitted to:

Air Force Office of Scientific Research
Holling Air Force Base
Washington, D. C. 20332

Attention: Lt. Col. Robert W. Carter

Submitted by:

Rafael M. Inigo
Associate Professor

Eugene S. McVey
Professor

Chen Ho Hsin
Graduate Student

Jay I. Minnix
Graduate Student

Chiewdarn Narathong
Graduate Student

Valerie Davis
Graduate Student

Report No. UVA/525647/EE86/101

February 1986

ERIC FILE COPY

Approved for public release;
distribution unlimited.



COMPUTER ENGINEERING AND CONTROL LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCES
UNIVERSITY OF VIRGINIA



86 6 10 127

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 86-0282	2. GOVT ACCESSION NO. AD-A168521	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Biological Visual Systems Structures for Machine Vision Applied to Robotics		5. TYPE OF REPORT & PERIOD COVERED Final Report
7. AUTHOR(s) R. M. Inigo, C. H. Hsin, C. Narathong, E. S. McVey, J. I. Minnix, V. Davis		6. PERFORMING ORG. REPORT NUMBER UVA/525647/EE86/101
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering University of Virginia, Thornton Hall Charlottesville, VA 22901		8. CONTRACT OR GRANT NUMBER(s) AFOSR-84-0349
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NE Bolling Air Force Base, Building 410 Washington, D.C. 20332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 230584
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE + Block Report Coverage 9/15/84 - 1/31/86
		13. NUMBER OF PAGES
		15. SECURITY CLASS. (of this report) Unclass
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution is limited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Human Visual system Machine visual sensor.,		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the research on a biological visual system (BVS) based sensor with possible applications to robotics and automation. The research covers the period October 1984 to January 1985 and was performed by two principal investigators, a participant faculty researcher and four graduate students working during various time periods.		

UNCLASSIFIED

The report covers the following subjects:

- (a) sensor configuration
- (b) edge detection modelling for the human visual system and edge detection using the BVS sensor
- (c) qualitative motion detection using the BVS
- (d) target tracking algorithms for the BVS
- (e) microsaccadic eye movement in the human visual system (HVS).

These five subjects, in the indicated order, constitute the five chapters of the report.

In Chapter I, three different configurations for the image plane are considered: "rectangular elements" (actually arcs of rings), circular elements and hexagonal elements. Rectangular elements consist of arcs of rings with exponentially increasing diameter which map to a true rectangular array in computation plane. The circular element configuration consists of concentric rings each with the same number of circular elements increasing exponentially in diameter which map to computation plane as columns of circles, all of the same size. The hexagonal element array is similar to the circular case, but with hexagonal elements mapping to columns of equal size hexagons in the computation plane.

Fundamental properties of the peripheral HVS are discussed in Chapter II, especially those relevant to edge detection. A mathematical model for HVS system edge detection is discussed in detail and a simplified but accurate mathematical expression of this model is used for edge detection simulation using real images. Algorithms for edge detection for the BVS sensor with hexagonal and circular elements are developed and applied to real images. The results are very similar to those obtained with the models proposed for the HVS.

In Chapter III the properties of the logarithmic conformal transformation are used to analyze edge images of various objects and to determine properties that allow estimation of the direction of motion for the following cases: scaling (magnification) and rotation about the optical axis (OA) of objects enclosing and not enclosing the OA; translation of objects enclosing the OA.

Chapter IV considers the quantitative estimation of target perturbation using two different techniques. In the first one the difference of the logarithmic function between two consecutive frames is linearized and estimated values for the displacement are obtained using the minimum mean square error. The second technique, based in the intensity function, does not require a solution to the correspondence problem and a relatively simple recursive algorithm is developed to estimate the displacement vector. However, due to the nonuniform sampling (in other words the different pixel size) of the BVS sensor, a point with a given intensity in a frame will not correspond to a point with the same intensity in the next frame even when the image would have the same illumination in both frames.

In Chapter V, the hypothesis that microsaccades sweep visual stimuli across the several receptors in the fovea is pursued. This hypothesis is supported by other researchers. If the hypothesis is true, fault tolerant sensor systems can be developed because of inoperative "pixel" output can be replaced with a combination of its neighbors. The technique could also be used for local sensor calibration.

UNCLASSIFIED

Final Report
on Grant No. AFOSR-84-0349

**BIOLOGICAL VISUAL SYSTEMS STRUCTURES
FOR
MACHINE VISION APPLIED TO ROBOTICS**

submitted to
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, D. C. 20332
Attention: Lt. Col. Robert W. Carter

submitted by

Rafael M. Inigo
Associate Professor

Eugene S. McVey
Professor

Chen Ho Hsin
Graduate Student

Jay I. Minnix
Graduate Students

Chiewdarn Narathong
Graduate Student

Valerie Davis
Graduate Student

Report No. UVA/525647/EE86/101

February 1986



**Approved for release by the
distribution committee.**

A-1
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
AFOSR-84-0349
Final Report
February 1986
Machine Vision Applied to Robotics

PREFACE

This report describes the research on a biological visual system (BVS) based sensor with possible applications to robotics and automation. The research covers the period October 1984 to January 1985 and was performed by two principal investigators, a participant faculty researcher and four graduate students working during various time periods.

The report covers the following subjects:

- a) sensor configuration
- b) edge detection modelling for the human visual system and edge detection using the BVS sensor
- c) qualitative motion detection using the BVS
- d) target tracking algorithms for the BVS
- e) microsaccadic eye movement in the human visual system (HVS).

These five subjects, in the indicated order, constitute the five chapters of the report.

The fundamental properties of a sensor emulating the configuration of the HVS sensor have been discussed previously by other researchers and are reviewed briefly in pertinent sections of Chapters I and II. It is convenient, however, to define the basic nomenclature and notation at this point.

The light sensitive array of elements equivalent to the retina in the BVS sensor is named the "image plane" and described mathematically by a complex plane,

$$z = x + jy$$

It is believed that the information transmitted from the retina to the cerebral cortex in the BVS undergoes some preprocessing before reaching the cerebral cortex. However, in the proposed sensor we assume direct mapping from the image plane to the computation plane, also a complex plane

$$w = u + jv$$

The mapping is a conformal transformation given by

$$\begin{aligned} w &= \ln(z) \\ &= \ln(re^{j\phi}) \\ &= \ln r + j\phi \end{aligned}$$

In other words,

$$u = \ln r$$

where r is the norm of z , and

$$v = \phi$$

where ϕ is the phase of z .

The above relations show that if an image is magnified with respect to the optical axis, its shape in computation plane remains invariant with only a shifting along the real axis u .

Similarly, if an image is rotated with respect to the optical axis in image plane, its shape remains invariant in computation plane with a shifting along the imaginary axis v .

The fact that we are assuming direct mapping from image plane to computation plane does not mean that in the course of further research we could not consider preprocessing operations taking place between the two planes.

In Chapter I, three different configurations for the image plane are considered: "rectangular elements" (actually arcs of rings), circular elements and hexagonal elements. Rectangular elements consist of arcs of rings with exponentially increasing diameter which map to a true rectangular array in computation plane. The circular element configuration consists of concentric rings each with the same number of circular elements increasing exponentially in diameter which map to computation plane as columns of circles, all of the same size. The hexagonal element array is similar to the circular case, but with hexagonal elements mapping to columns of equal size hexagons in the computation plane. Design rules are developed relating the number of elements per ring to the number of rings and "fovea" size. The fovea is the small circular region at the center of the image plane where, for the moment, no elements are placed. Advantages and disadvantages of each configuration are discussed and simulations are performed using images obtained with a conventional camera. The savings in processing time stemming from the enormous reduction in the number of pixels is quantitatively discussed.

Fundamental properties of the peripheral HVS are discussed in Chapter II, especially those relevant to edge detection. A mathematical model for HVS system edge detection is discussed in detail and a simplified but accurate mathematical expression of this model is used for edge detection simulation using real images. Algorithms for edge detection for the BVS sensor with hexagonal and circular elements are developed and applied to real images. The results are very similar to those obtained with the models proposed for the HVS.

In Chapter III the properties of the logarithmic conformal transformation are used to analyze edge images of various objects and to determine properties that allow estimation of the direction of motion for the following cases: scaling (magnification) and rotation about the optical axis (OA) of objects enclosing and not enclosing the OA; translation of objects enclosing the OA. The algorithm works for both geometric and random figures and its automatic implementation by means of a computer program that will indicate the type of motion is now in progress. The algorithm needs to be extended to include translation of objects not enclosing the OA.

Chapter IV considers the quantitative estimation of target perturbation using two different techniques. In the first one the difference of the logarithmic function between two consecutive frames is linearized and estimated values for the displacement are obtained using the minimum mean square error. This technique works well and simulations confirm the accuracy of the estimation. However it presupposes that the correspondence problem has been solved, i.e., that it is known that two points correspond to each other in the two images. The second technique, based in the intensity function, does not require a solution to the correspondence problem and a relatively simple recursive algorithm is developed to estimate the displacement vector. However, due to the nonuniform sampling (in other words the different pixel size) of the BVS sensor, a point with a given intensity in a frame will not correspond to a point with the same intensity in the next frame even when the image would have the same illumination in both frames. Consequently, the algorithm as developed in Chapter IV does not perform adequately in simulations. Its modification to take into account the nonuniform sampling

is currently under consideration.

In Chapter V, the hypothesis that microsaccades sweep visual stimuli across several receptors in the fovea is pursued. This hypothesis is supported by other researchers. If the hypothesis is true, fault tolerant sensor systems can be developed because an inoperative "pixel" output can be replaced with a combination of its neighbors. The technique could also be used for local sensor calibration. This is the only chapter in which, for the moment, the fovea region has been considered. Further research on the BVS sensor will require consideration of the fovea region which has a uniform distribution of elements, in addition to the region outside the fovea considered in this report, with its log spiral distribution of elements. The role played by the fovea, as well as previously unexpressed characteristics of the external region, will have to be considered and included in the BVS sensor.

TABLE OF CONTENTS

CHAPTER I

I.1	Introduction	1
I.2	Literature Search	7
I.3	Geometric Considerations	11
I.4	Simulations	27
I.5	Verification of Properties	36
I.6	Comparison to a Conventional Sensor	41
I.6.1	Resolution	43
I.6.2	Number of Elements	44
I.6.3	Field of View	45
I.7	Conclusions	47
	References	48

CHAPTER II

II.0	Introduction	51
II.1	Properties of the Peripheral Visual System	52
II.1.1	Retinal Ganglion Cells	52
II.1.2	Lateral Geniculate Nucleus	55
II.1.3	Distribution and Shape of the Receptive Field	56
II.2	Experiment on Peripheral Vision	61
II.2.1	Results	64
II.3	Simulation of Edge Detection in the Human Visual System	65
II.3.1	Scheme of the Geometric Distribution	65

II.3.2	Weighting Functions of the Receptive Field	66
II.3.3	Convolution	67
II.3.4	Simulation Procedure and Results	69
II.4	Edge Detection Simulation for New Sensor	86
II.4.1	Simulation Procedure and Results	89
II.5	Savings in Computations for New Sensor Compared to Rectangular Sensor	89
II.6	Comments and Conclusions	90
	References	103
CHAPTER III		
III.1	Introduction	106
III.2	Magnification and Rotation	107
III.2.1	Object Edges Enclosing the Optical Axis	107
III.2.2	Objects with Edges not Enclosing the Optical Axis	112
III.2.3	Multiple Objects	120
III.3	Translation Along a Straight Line in Image Plane	124
III.4	Conclusions and Future Plans	140
	Reference	143
CHAPTER IV		
IV.1	Introduction	144
IV.2	Linear Estimation of Target Perturbations Utilizing the Logarithmic Function	144
IV.2.1	Translational Case	145

IV.2.2 Translation, Rotation and Scaling	148
IV.3 Computer Simulation	155
IV.4 A Tracking Algorithm Using the Intensity Function	157
IV.5 Conclusion and Plans for the Future	161
References	167
CHAPTER V	168
APPENDIX A	
A.1 Programs Listing for Chapter I	A-1
A.2 Programs Listing for Chapter II	A-2
A.3 Programs Listing for Chapter III	A-3
A.4 Programs Listing for Chapter IV	A-4
APPENDIX B	
Detail Structure Configurations	B-1

LIST OF FIGURES

- Fig I.1 Schultze's Model of the Retina
- Fig I.2 Sandini and Tagliasco's Model of the Retina
- Fig I.3 Example of form invariance under magnification
- Fig I.4 Example of form invariance under rotation
- Fig I.5 Image and computational planes for rectangular elements
- Fig I.6 Example of different inter-ring spacing constants
- Fig I.7 Image and computational planes for circular elements
- Fig I.8 Relationship between adjacent circular elements
- Fig I.9 Computational plane relationship for circular elements
- Fig I.10 Image and computational planes for hexagonal elements
- Fig I.11 Angle relationships for hexagonal elements
- Fig I.12 Relationship between adjacent hexagonal elements
- Fig I.13 Original images
- Fig I.14(a) Rectangular element simulation—form invariance under magnification
- Fig I.14(b) Rectangular element simulation—form invariance under rotation
- Fig I.14(c) Rectangular element simulation—distortion due to shifting
- Fig I.14(d) Rectangular element simulation—resolution of simulation
- Fig I.15(a) Circular element simulation—form invariance under magnification
- Fig I.15(b) Circular element simulation—form invariance under rotation
- Fig I.15(c) Circular element simulation—distortion due to shifting
- Fig I.15(d) Circular element simulation—resolution of simulation
- Fig I.16 Circular element computational plane—uncovered sensor area
- Fig I.17 Circular element image plane—uncovered sensor area 71 by 32 Sampling Points
- Fig II.1(a) The human visual system
- Fig II.1(b) Relationship of peripheral visual system with retina and visual cortex
- Fig II.2 The different neuron layers in the vertebrate retina
- Fig II.3 The profiles of the four mechanisms
- Fig II.4(a) Measuring angle of detection for sample motion on the XY plane
- Fig II.4(b) Measuring angle of detection for sample motion on the XZ plane
- Fig II.5 Block diagram of preprocessing in human visual system

- Fig II.6 The scheme of the geometric centers of the receptive fields
- Fig II.7 The exact and the approximated DOG functions
- Fig II.8 (a) Change of intensity in two dimensions (b) Zero-crossing for a radial cross section
- Fig II.9(a) Geometry of sampling grid (concentric rings)
- Fig II.9(b) Geometry for simulation using rectangle pixels
- Fig II.10 71 by 32 sampling points
- Fig II.10(a) Original image: white square on black background
- Fig II.10(b) Image after edge detection in image plane: bright points are positive values; dark points are negative values
- Fig II.10(c) Zero-crossing points (real edges) in image plane
- Fig II.10(d) Real edges in computation plane
- Fig II.11 Edge detection simulation using HVS detection model for 113 by 51 sampling points
- Fig II.11(a) Image after edge detection in image plane: bright points are positive values; dark points are negative values
- Fig II.11(b) Image after edge detection in computational plane
- Fig II.11(c) Zero crossing points (real edges) in image plane
- Fig II.11(d) Real edges in computation lane
- Fig II.12 Edge Detection Simulation Using HVS Detection Model For 71 by 32 Sampling Points.
- Fig II.12(a) Original image: noisy photograph
- Fig II.12(b) Image after edge detection, image plane
- Fig II.12(c) Image after edge detection, computation plane
- Fig II.12(d) Zero crossing edges, image plane
- Fig II.12(e) Zero crossing edges, computation plane
- Fig II.13 Edge Detection Simulation Using HVS Detection Model for 113 by 51 Sampling Points
- Fig II.13(a) Image after edge detection, image plane
- Fig II.13(b) Image after edge detection, computation plane
- Fig II.13(c) Zero crossing edges, image plane, no thresholding
- Fig II.13(d) Zero crossing edges, image plane, thresholding = 20
- Fig II.13(e) Zero crossing edges, computation plane, no thresholding
- Fig II.13(f) Zero crossing edges, computation plane, thresholding = 20
- Fig II.14 Block diagram of image processing procedure for the new sensors
- Fig II.15 Circular elements and hexagonal elements neighborhoods in computation plane
- Fig II.16 Block diagram of the Laplacian operation

- Fig II.17 Block diagram of the general edge detection scheme
- Fig II.18 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings 74 Elements per Ring)
- Fig II.18(a) Original image: white square on black background
- Fig II.18(b) Mapping of image to computation plane
- Fig II.18(c) Edge detection in computation plane by absolute value method and no thresholding
- Fig II.18(d) Edge detection in CP by absolute value method; threshold = 120
- Fig II.18(e) Edge detection in CP by Laplacian method, no thresholding
- Fig II.18(f) Edge detection in CP by Laplacian-Gaussian method, no thresholding
- Fig II.19 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring)
- Fig II.19(a) Original grey level image: dark frame on light background
- Fig II.19(b) Mapping of image to CP
- Fig II.19(c) Edge detection in CP by absolute value method; threshold = 190
- Fig II.19(d) Edge detection in CP by Laplacian method; threshold = 30
- Fig II.19(e) Edge detection in CP by Laplacian-Gaussian method, no thresholding
- Fig II.20 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).
- Fig II.20(a) Original image
- Fig II.20(b) Simulated circular-element sensor image
- Fig II.20(c) Mapping of image to computation plane
- Fig II.20(d) Edge image in CP, absolute value method, threshold = 75
- Fig II.20(e) Edge image in CP, Laplacian method, threshold = 20
- Fig II.20(f) Edge image in CP, Laplacian-Gaussian method, no thresholding
- Fig II.21 Edge Detection Using Simulation of Circular-Elements Sensor (55 Rings, 113 Elements per Ring).
- Fig II.21 Edge Detection Using Simulation of Circular-Elements Sensor (55 Rings, 113 Elements per Ring).
- Fig II.21(a) Simulated circular-element sensor image
- Fig II.21(b) Mapping of image to computation plane
- Fig II.21(c) Edge image in CP, absolute value method, threshold = 55
- Fig II.21(d) Edge image in CP, Laplacian method, threshold = 15
- Fig II.21(e) Edge image in CP, Laplacian-Gaussian, no thresholding
- Fig III.1 Random figure in image plane
- Fig III.1(a) Random figure edge in computation plane

- (A) Original; (B) Figure magnified by $m=2$ about optical axis in IP
- Fig III.1(b) Random figure edge in computation plane
(A) Original; (B) Figure rotated by $+45^\circ$ about optical axis in IP
- Fig III.2 Ellipse: (a) Original, A and magnified by 2.3, B; (b) Original, A and scaled by 0.8, B.
- Fig III.3 Random figure: (a) Original, A and rotated by $+30^\circ$, B; (b) Original, A and rotated by -30° , B.
- Fig III.4 Random figure edge in CP: (A) Original; (B) Magnified by $m=2$ and rotated by $\alpha=+45^\circ$ both about optical axis
- Fig III.5 Random figure edge in CP: (A) Original; (B) Translated by $\mu=0.3$ in IP.
- Fig III.6 Mapping of off-center, by (4,4) units, random figure.
(A) $m=1$; (B) $m=0.8$
- Fig III.7 Mapping of off-center, by (4,4) units, random figure.
(A) $m=0.8$, (B) $m=1$, (C) $m=1.2$
- Fig III.8 Mapping of off-center, by (4,4) units, random figure. (A) $m=1$; (B) $m=1.2$
- Fig III.9 Mapping of off-center, by (4,4) units, random figure. (A) $m=1$, $\alpha=0$; (B) $m=1$, $\alpha=+30^\circ$
- Fig III.10 Mapping of off-center, by (4,4) units, random figure.
(A) $m=1$, $\alpha=0$; (B) $m=1.2$, $\alpha=20^\circ$
- Fig III.11 Mapping of off-center, by (4,4) units, random figure.
(A) $m=1$, $\alpha=0$; (B) $m=2$, $\alpha=30^\circ$
- Fig III.12(a) Centered square and off-center rectangle in image plane
- Fig III.12(b) Mapping of the two objects in Fig 12(a) to the CP.
- Fig III.12(c) Mapping of Fig 12(a) to CP.
(A) Original; (B) $m=1$, $\alpha=+15^\circ$
- Fig III.12(d) Mapping of Fig 12(a) to CP.
(A) Original; (B) $m=1.2$, $\alpha=0$
- Fig III.13 Mapping of Fig 12(a) to CP.
(A) Original; (B) $m=1.2$, $\alpha=45^\circ$
- Fig III.14 Circle with $r=6$; edge in CP.
(a) (A) Centered; (B) off-center by (0.8, 0.6) in IP
(b) (A) Centered; (B) off-center by (-0.6, -0.8) in IP
- Fig III.15 Square with side $=2$; edge in CP.
(a) (A) Centered, (B) off-center by (-0.4, 0.5) in IP
(b) (A) Centered, (B) off-center by (0, 0.5) in IP

- (c) (A) Centered, (B) off-center by (0.5, -0.4) in IP
- Fig III.16 Ellipse with semiaxis $a=4$, $b=3$; edge in CP.
(a) Centered; (b) off-center by (1,-1) in IP
- Fig III.17 Random figure edge in CP.
(a) Centered; (b) Translated by (0, 0.5) in IP
- Fig III.18 Random figure edge in CP.
(a) Centered; (b) Translated by (0, -0.5) in IP
- Fig III.19 Random figure edge in CP.
(a) Centered; (b) Translated by (0.3, 0.5) in IP
- Fig III.20 Random figure edge in CP.
(a) Centered; (b) Translated by (0.3, -0.5) in IP
- Fig III.21 Random figure edge in CP.
(a) Centered; (b) Translated by (-0.3, 0.5) in IP
- Fig III.22 Random figure edge in CP.
(a) Centered; (b) Translated by (-0.3, -0.5) in IP
- Fig III.23 Off-center random object moving on a line at 45° ; B displaced by (1,1) with respect to A.
- Fig III.24 Off-center random figure moving (a) and a line at 60° with A at (4, 6.93) and B at (5, 8.66); (b) on a line at 120° with C at (-4, 6.93) and D at (-5, 8.66)
- Fig IV.1(a) A horizontal perturbation ($\Delta\hat{y}$) of a rectangle 4 by 6 (Translation case)
- Fig IV.1(b) A vertical perturbation ($\Delta\hat{y}$) of a rectangle 4 by 6 (Translation case)
- Fig IV.2(a) A horizontal error ($\Delta x - \Delta\hat{x}$) for Fig IV.1(a)
- Fig IV.2(b) A vertical error ($\Delta y - \Delta\hat{y}$) for Fig IV.1(b)
- Fig IV.3(a) A horizontal perturbation ($\Delta\hat{x}$) of a rectangle 4 by 6 (Translation, Rotation and Scaling case)
- Fig IV.3(b) A vertical perturbation ($\Delta\hat{y}$) of a rectangle 4 by 6 (Translation, Rotation and Scaling case)
- Fig IV.4(a) A horizontal error ($\Delta x - \Delta\hat{x}$) for Fig IV.3(a)
- Fig IV.4(b) A vertical error ($\Delta y - \Delta\hat{y}$) for Fig IV.3(b)
- Fig IV.5(a) A horizontal error ($\Delta x - \Delta\hat{x}$) versus number of sampling points
- Fig IV.5(b) A vertical error ($\Delta y - \Delta\hat{y}$) versus number of sampling points

LIST OF SYMBOLS

1. A 2×2 homogeneous affine transformation matrix
2. \hat{A} 2×2 affine perturbation matrix
3. $A(R)$: Gain constant of the weighting function
4. \underline{a} 6×1 or 4×1 vector of homogeneous affine transform parameters
5. a 4×1 vector of algorithm estimate of \underline{a}
6. a_c 2×1 vector of constrained affine transform parameters
7. \underline{b} 2×1 affine transform translation vector
8. D 2×1 displacement vector
9. \hat{D} 2×1 algorithm estimate of D
10. E 2×1 error vector
11. $f(R, \phi)$: The input image which is formed in the retina
12. $f(x, y)$ Logarithmic function, as defined in text
13. G $N \times 2$ complex matrix of spatial partial derivatives
14. $g(R, \phi)$: The output image
15. I 2×2 Identity matrix
16. $I(w, t)$ The intensity values of an image in computation plane.
17. $\text{Im}()$ Imaginary part of a complex number
18. i, k Integer indices
19. $J(x, t)$ The intensity values of an image in image plane
20. $j = \sqrt{-1}$

21. m number of sampling points
22. P $N \times 2$ complex matrix of weighted spatial derivatives
23. P : Spatially variant constant
24. R : Eccentricity
25. $\text{Re}()$ Real part of a complex number
26. r : The radius of a receptive field
27. \underline{x} 2×1 vector
28. α The partial derivative of $f(x,y)$ with respect to x
29. $\bar{\alpha}$ Complex conjugate of α
30. β The partial derivative of $f(x,y)$ with respect to y
31. $\bar{\beta}$ Complex conjugate of β
32. $\Delta T(x,t)$ $N \times 1$ frame difference vector
33. $\underline{\Delta T}$ Complex conjugate of ΔT
34. $\nabla()$ Denotes the gradient
35. ΔT The frame difference (complex quantity)
36. ∇^2 : Laplacian operator, $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$
37. θ Angle of rotation
38. κ Scaling factor
39. σ : The standard deviation
40. τ The time between two successive frames
41. $w(R,r)$: The weighting function of receptive fields

42. $\| \cdot \|$ Norm of a vector

GLOSSARY

Computational Plane (CP) - the plane onto which the image is mapped under the conformal logarithmic transformation.

Current Image (CI) - the second (most recent) image in a two-frame image sequence.

Difference Image (DI) - the difference between the current image and the previous image, in a two-frame image sequence, obtained in the computation plane.

Difference of Gaussians (DOG) - a function consisting of the difference of two Gaussian distributions. It is thought that DOG functions describe the shape of the retinal ganglion cells receptive fields.

Eccentricity - in the human visual system, the distance from the center of the fovea to the center of a receptive field; in an artificial sensor, the distance from the optical axis, i.e. the center of the image, to a point or pixel in the image. Usually expressed in degrees.

Eight-neighborhood - area of an image in which the pixel under consideration is surrounded by eight other pixels. It is used in conventional rectangular arrays.

Field of view (FOV) - the slide angle covered by the lens system of a vision system; in cameras, the part of the scene projected to the light-sensitive sensor.

Fovea - the central part of the retina in which cones are densely packed and no rods are present; it is the area where visual acuity in humans is greatest. In an artificial sensor, the circular part of the center where receptive fields do not follow the log-spiral configuration.

Frame - one image in a sequence obtained by a camera (TV or movies).

Frame difference - the difference between the images of two frames, not necessarily two consecutive frames.

Gaussian (G) - the normal probability distribution function, $p(x) = (1/\sqrt{2\pi})e^{-x^2/\sigma^2}$.

Image Plane (IP) - the plane where the actual image is formed; the retina in the eye, the film in a photographic camera, and the CCD array in a solid state TV camera.

Inter-ring spacing - the exponentially increasing space between rings of elements in the image plane of the BVS sensor.

Laplacian - a mathematical operation representing the divergence of the gradient of a scalar quantity; mathematically, in two dimensions, in rectangular cartesian coordinates, the Laplacian operator is $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$.

Lateral Geniculate Nucleus (LGN) - the main visual nucleus between the eye and the brain. It is fed by the optic nerve, which consists of axons of the retinal ganglion cells. The axons emerging from the LGN, called the optic radiations, project to the striate cortex in man.

Motoneurons - neurons which carry impulses controlling muscle contractions.

Myocardial - pertaining to the muscles or muscle tissue.

Number of points - the number of discrete points, or pixels, resulting from the difference between the current and the previous images in binary form.

Periphery - the outer part of the retina or of the image plane, where the log-spiral structure is found.

Pixel - a term formed from the words picture-element; it means each one of the discrete light sensitive elements in the sensor of a CCD camera, or the quantized elements formed from the scan lines of a vidicon camera.

Pixel fitting - the process of fitting small rectangular pixels into larger pixels of a different shape for simulation purposes.

Previous image - in a sequence of two consecutive frames, the first of the two frames, followed by the 'current image' or second frame.

Reticulate formation - a portion of the brain containing elements which play a role in the formation of conditioned reflexes, the regulation of sensory input, and consciousness.

Retinal ganglion cells - the final layer of cells in retinal processing.

Retinotopic - concerning the arrangement of receptive fields on the retina; in the peripheral field, the resolution decreases with eccentricity.

Rigid motion - a simple type of motion in which all the objects in a scene move with the same velocity.

Scotoma - a blind spot in the retina due to disease or injury (not the "blind spot").

Six-neighborhood - area of an image in which the pixel under consideration is surrounded by six other pixels. It is found in hexagonal or circular grid

configurations.

Striate cortex - the primary cortical receiving area in man.

Template matching - a pattern recognition technique in which an image is compared to a template stored in memory (in digital form) for identification purposes.

X cells and Y cells - the two classes of retinal ganglion cells.

Zero-crossing - point at which a function's value changes its sign.

CHAPTER I. ANALYSIS OF GEOMETRIC CONFIGURATIONS

I.1. Introduction

One of the major problems with practical machine vision systems is the time involved in obtaining and processing images. The objective of this research project is to use several properties of biological vision systems (BVS) in the design of a new sensor which will reduce processing time in many machine vision applications. Current results indicate that these novel ideas will indeed be useful in a variety of industrial, scientific, and military applications.

There are three basic features of the eye that are being investigated for application to the new machine vision system. The first of these is the actual structure of the retina, the second is the preprocessing in hardware, and the third is the conformal mapping into the computational space in the cerebral cortex. Each of these has characteristics that will make some tasks performed by machine vision systems easier or faster.

For many years, the organization of light sensitive receptors in the retina has been known. Over a hundred years ago, Shultze [1] discovered that the receptive fields in the retina increase linearly in diameter with increasing eccentricity, Fig. I.1. Later studies [2] confirm this early work and help to demonstrate that an image formed on the retina undergoes a complex logarithmic conformal mapping into the cerebral cortex, as a result of the geometric configuration of the receptive fields.

This structure allows usable information to be obtained using a significantly smaller number of pixels (light sensitive elements) than that necessary in conventional sensors [3]. Since the elements in the retina

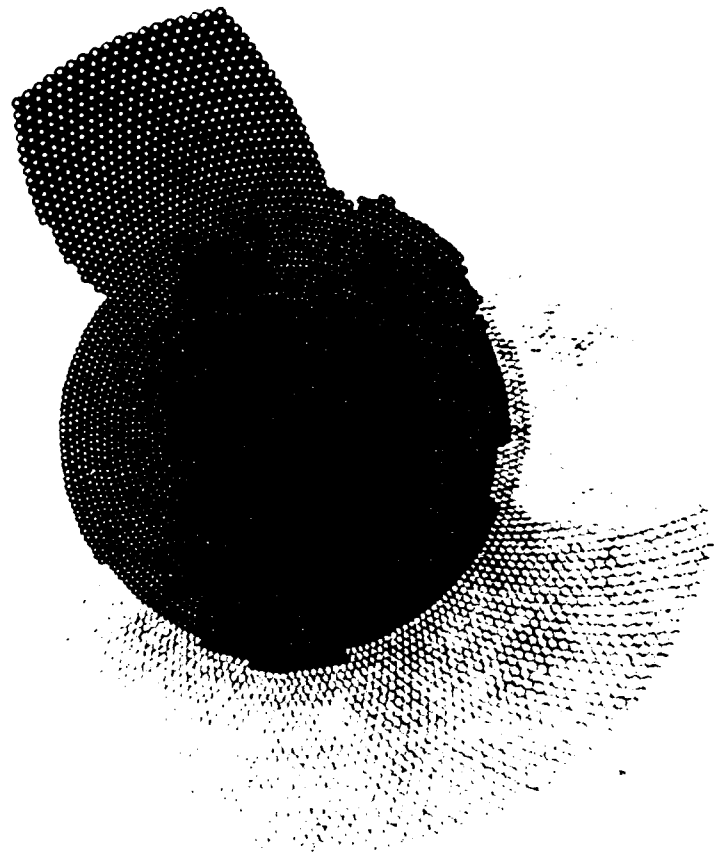


Figure I.1. Schultze's Model of the Retina.

increase in size with increasing radius, the outer elements are much larger than the inner ones; but, in a conventional sensor the elements are of uniform size regardless of location, causing a much greater number to be used, Fig. I.2. If a machine vision sensor is constructed that has a structure similar to that of the retina, a considerable savings will result in terms of the number of elements necessary to produce a given resolution near the center of the array. Since fewer elements are being used, any manipulation of the image data will be correspondingly faster.

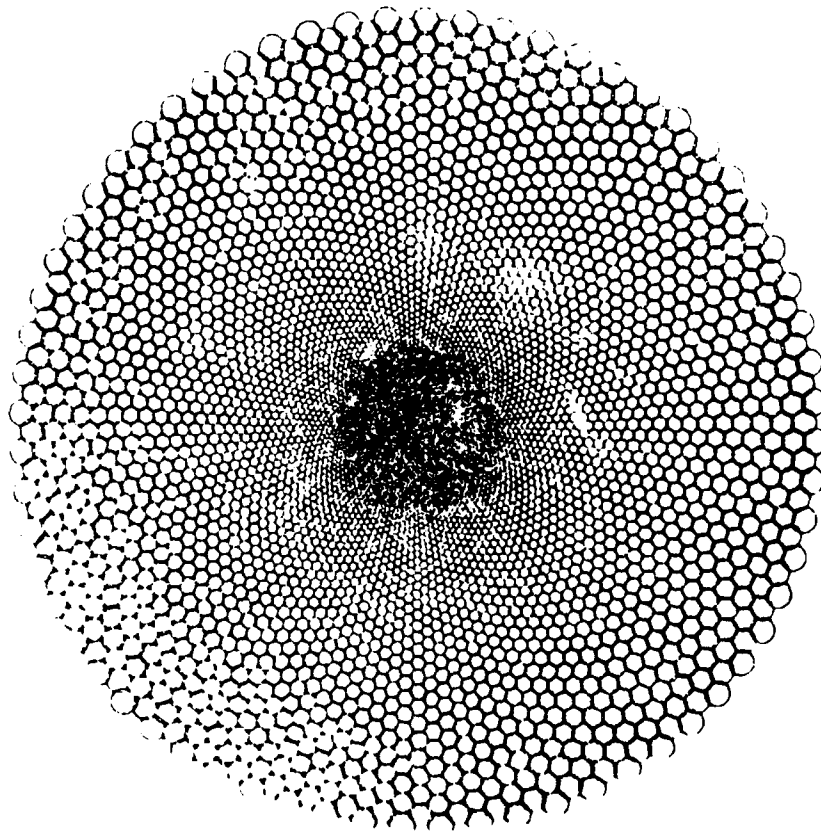


Figure I.2. Sandini and Tagliasco's Model of the Retina.

The second area of interest is the hardware preprocessing that is done in BVS between the retina and the cortex. It appears that many common image processing tasks are performed before the information reaches the brain in many biological systems [4]. The retinal interconnections between cells in the periphery of the retina may be useful in such image processing tasks as edge and motion detection [5].

Performing edge and motion detection in hardware as in BVS using the properties of the BVS sensor should produce a large increase in speed over a conventional software image processing scheme, and even over hardware systems based on the conventional sensor. Specialized hardware such as that in the eye could greatly reduce the complexity of now complicated situations, and make many machine vision applications much simpler.

The final aspect of BVS that we are interested in is the mapping that occurs between the retina and the cerebral cortex. Several studies [6] [7] have demonstrated that the arrangement of the receptive fields in the eye is transferred into the brain by means of a complex conformal mapping. This mapping, called the logarithmic spiral mapping (LSM), is circularly symmetric about the center of the eye and transforms radial lines of constant angle into horizontal lines in the computational plane, and circles about the center of the image plane into vertical lines placed according to the natural logarithm of the circle's radius in the computational plane [8].

The LSM exhibits several features that may be very useful in machine vision systems. The first of these is a form invariance under magnification, which means that if an object in the image plane is magnified about the optical axis, the shape of the object in the computational plane does not change, it only moves horizontally, Fig. I.3. The second feature is form invariance under rotation, which is similar to the previous one in that if the object is rotated about the optical axis in the image plane, the shape in the computational plane does not change, it only moves vertically, Fig. I.4. The last of these features is a built in resistance to noise that is due to the increasing size of the sensors in the periphery.

If these features can be properly used, they could greatly simplify tasks such as template matching or simple object identification. This would help to make many machine vision applications more feasible from a processing time standpoint [9], an important consideration for real time applications.

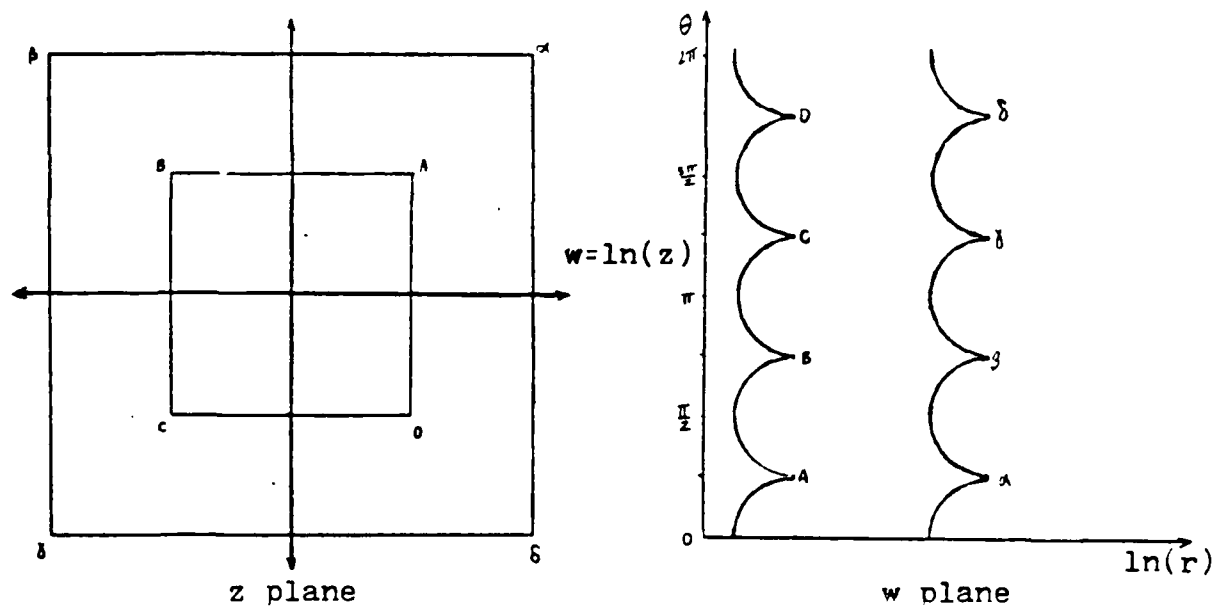


Figure I.3. Example of Form Invariance Under Magnification.

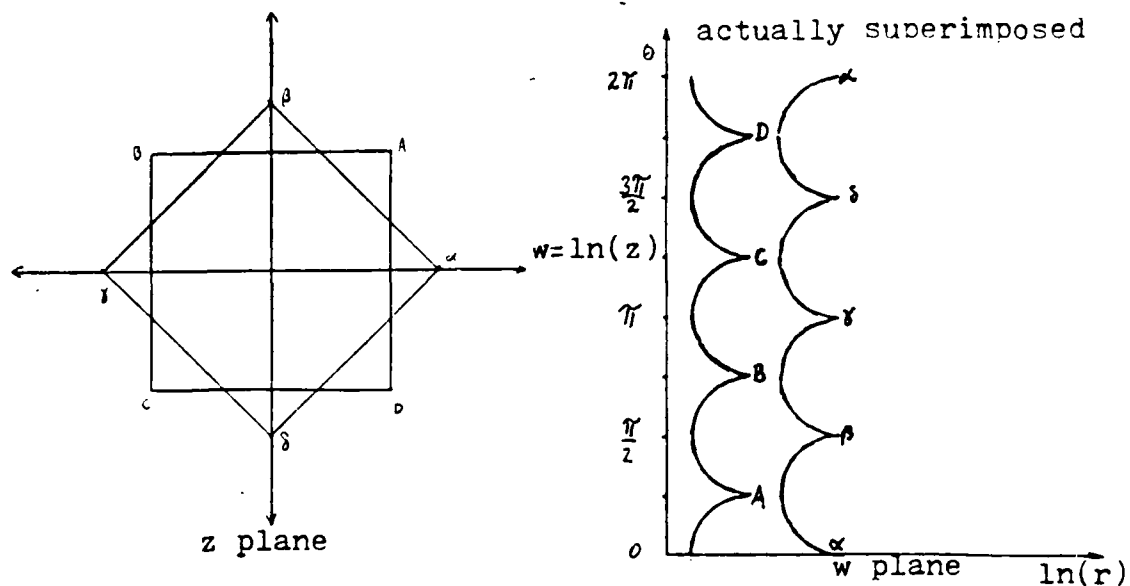


Figure I.4. Example of Form Invariance Under Rotation.

Given the diversity and quantity of basic research to be conducted, it was necessary to separate this project into several related parts, each to be conducted by different members of the research team. The area of investigation described in this chapter concerns the actual architecture of the sensor — to examine the structure and arrangement of the receptive fields (retina) and its mapping into the computational space, then to simulate this structure and test it to determine whether it behaves as predicted and can indeed be used as intended.

The work was divided into three interrelated areas, the first of which was an extensive literature search to provide a starting point from which to begin simulations of the biological vision system. This literature search provided valuable insight into previous work and allowed a decision to be made about the proper structures to simulate. The second aspect of this part of the project was a detailed analysis of the geometry of the structures intended for simulation, both the image plane configuration and the computationally mapped image. The final part was the actual simulation of the structures.

1.2. Literature Search

This section contains a discussion of the results of the literature search that was conducted prior to and during the other parts of the work. Important findings will be cited, and explanations about their usefulness will be provided.

The material in this area is broken up into four loose groups, each of which has a definite part in this project. The four classifications are physiology/psychology, engineering applications, hexagonal structures, and image sensor devices. In each group, articles and papers relevant to the research were obtained.

The physiological/psychological papers were fairly useful in increasing the investigator's understanding of the function and organization of the retina, the connecting structures, and the cortex. Starting with the paper by Schultze [1], there was a good deal of information in this area.

Schwartz has published several papers [6], [10] about the features of biological vision systems, including the characteristics in which we are most interested. His papers seem to support the direction that we have assumed, and provide a general physiological basis for our work. He is involved with research into the retinotopic mapping and the form invariance properties.

Many of the most important physiological/psychological papers were cited in Section I.1., such as the paper by Daniel and Whitteridge [2], which supports a great deal of the work done by Schwartz and others. Another good general paper is the one by Michael [5], which gives a basic description of the biological vision system and its functions. The paper by Lettvin, et al [4] was of use in understanding the processes that can be done in hardware before sending the information to the brain.

Unfortunately, an engineering oriented education leaves little time for physiology or psychology, so many of the articles that may have been helpful were so medically oriented that no useful information could be gained from them. The engineering applications papers were quite a bit more helpful.

The most important of the engineering papers have already been cited, particularly the papers by Sandini and Tagliasco [3] and Braccini, et al [9]. There is a second paper by Braccini's group [11], that supports and builds on the previous material. The Weiman and Chaikin [7], [8] papers were essentially similar to the ones above, and there is a third [12] that is based on [7] and [8]. The Weiman and Chaikin material supports the idea of a machine vision system based on the biological vision system. Another

engineering oriented paper by Jain [13], suggests the same type of sensor as the others. Finally, the work of Messner and Szu [14], [15], and [16] also contributes to the knowledge in this area, although it contains relatively little new material.

The third group, hexagonal structures, is important because the arrangement of the elements in the image and computational planes can be hexagonal, in which case, the computational plane's elements are in a six neighborhood as opposed to the eight neighborhood found in standard rectangular arrays. This forms a hexagonal lattice, where normal image processing techniques do not apply. The hexagonal structures papers address the use of hexagonal structures in image processing.

A paper by Scholten and Wilson [17] discusses the use of hexagonal lattices in the chain coding of line images. Their findings indicate that not only are hexagonal lattices useful, but they consistently outperform conventional rectangular grids. Golay [18] found that hexagonal grids were very useful for some applications, but he had doubts about their use in situations where the orientation of the image was important. Yajima [19] and his group demonstrated that Golay's doubts were unfounded, because they achieved approximately 60% data compression of oriented images using a similar structure. Others doing similar work are Burt [20] and Mersereau [21], whose findings indicate that the hexagonal lattice will be at least as good as a rectangular lattice, thus reinforcing the usefulness of the proposed sensor.

The last category, image sensor devices, was investigated to gain knowledge about what technology was available, and what would be the best choice for our sensor. Since General Electric has done a great deal of this type of research, several papers about their technology were available. A paper by Sachs [22] gives an overview of the devices currently in use in industry. Charge Injection Devices are discussed in a paper by Carbone and Hunter [23] and Carbone also has a paper on solid state sensors in general [24].

The literature indicates that there is a wide variety of possible technologies available. Unfortunately, it seems that the overwhelming majority of cameras today have square or rectangular light sensitive elements, whereas the proposed sensor would have either arc-of-ring, circular, or hexagonal elements. This is not a problem with the actual construction of the sensor, but it may be that existing computer aided design tools will not allow such shapes when designing the semiconductor masks.

The literature also shows that conventional sensors have straight line routings, either parallel or perpendicular only. The proposed sensor may require curved lines and routings that are no longer parallel or perpendicular. Again, this problem will be at its most severe during the design of the masks which, once complete, will allow the construction to proceed normally.

I.3. Geometric Considerations

There are a great number of questions to be answered about the geometry of the retina and the mapping. The basic idea is to build on the fundamental concepts found throughout the literature and construct several simulations of different structures to indicate which would best suit the design objective. From this starting point, it was fairly easy to decide that there were really only three feasible structures to consider, which are called rectangular elements, circular elements, and hexagonal elements. Each of these structures has advantages and disadvantages, and part of the purpose of this section is to discuss these advantages and disadvantages.

The geometry of these structures is important not only in itself, but also as a part of the simulation of these structures. It is important to realize that in order to evaluate the proposed structure without actually constructing it, it must be simulated using a normal rectangular array camera. In order to do this, the exact geometry of each element in the proposed structure must be known.

The first and simplest of the three possible structures is the rectangular element case, which consists of exponentially spaced concentric rings, each with the same number of elements. This image plane configuration maps into a rectangular array of elements in the computational plane, Fig. I.5.

This structure has many advantages, not the least of which is its simplicity. The concentric circles and radial lines make it very easy to simulate using a conventional rectangular array. Its design and construction are also simpler than the other two. Since lines and circles are easy to write

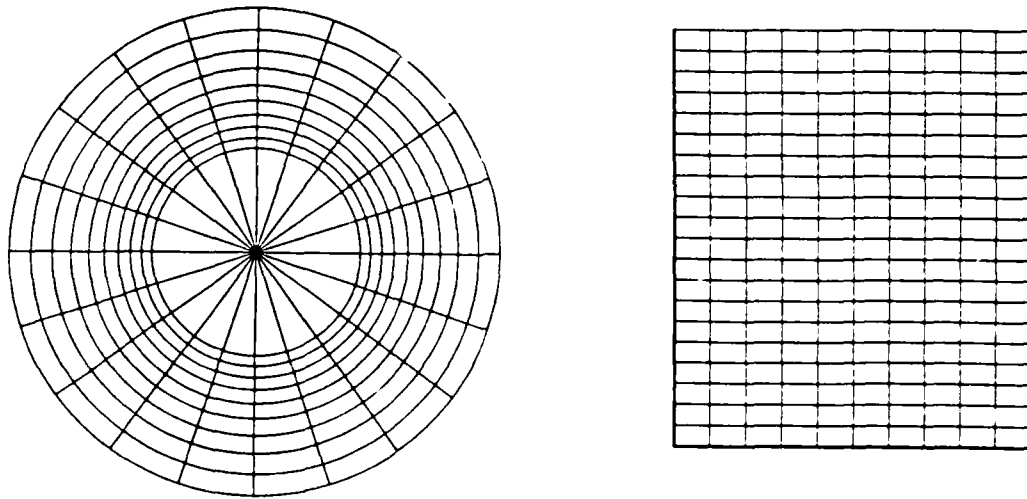


Figure 1.5. Image and Computational Planes . . . Rectangular Elements.

equations for, the problem of pixel fitting a square array onto this structure becomes correspondingly easier.

The fact that this architecture maps into a rectangular array under the LSM allows the use of existing image processing techniques. The Sobel operator for edge detection, for example, can be used to detect edges in the computational plane in the conventional way. Of course the results will be obtained much faster (fewer elements), and they will have vastly different meanings (horizontal lines in the computational plane mean radial lines in the image plane and vertical lines mean circles in the image plane), but no major

rethinking of image processing techniques will be necessary.

Rectangular elements have several design advantages over the other schemes. The shape of the elements is like that of existing sensors, whose elements are square or rectangular in shape. This may be of importance in the design and layout of the actual sensor. Both the number of elements and the inter-ring spacing can be controlled, allowing more selectivity in terms of resolution over the other two structures.

In the circular and hexagonal element cases the spacing between the rings is determined by the geometry of the sensor, while in the rectangular case it is up to the designer. This will allow the selection of the size change between rings, Fig. I.6. Another advantage of the rectangular elements is the

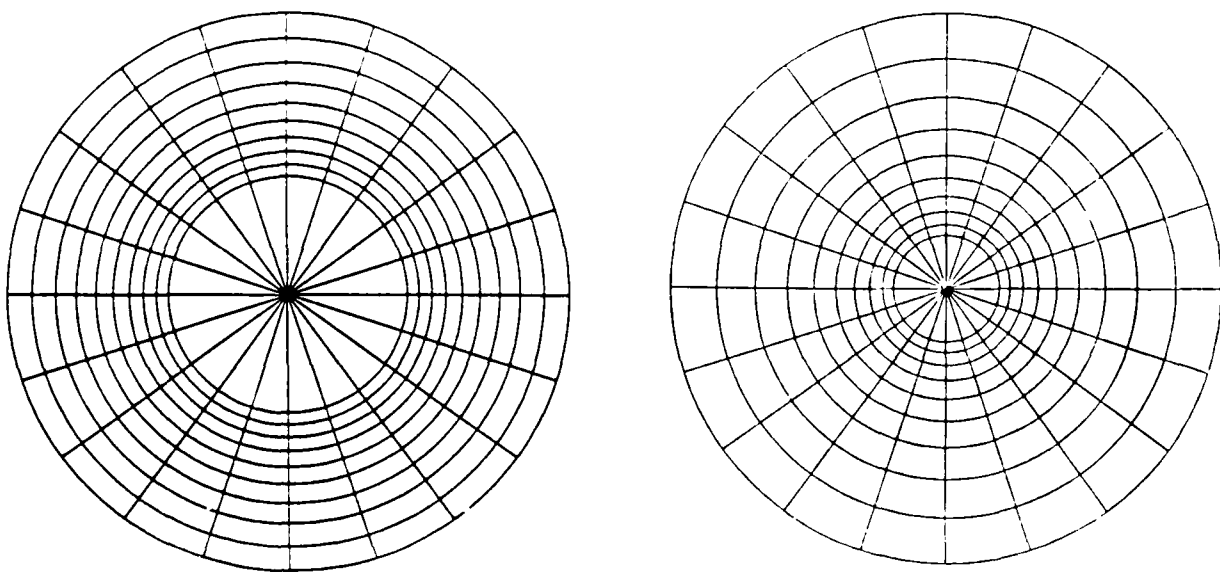


Figure I.6. Example of Different Inter-ring Spacing Constants.

straight lines running from the center of the array to the outside. In the actual sensor these may make line routings easier.

The only real disadvantage of this structure is that it is the least similar to the actual structure of the eye. Both of the other arrangements approximate better the actual retinal structure. What is meant by this is that the arrangement of the receptive fields in the retina more closely resembles either of the other two structures.

The second of the three possible structures is the circular element case, which consists of concentric rings of increasing radius, each with the same number of circular elements. Each ring is offset from its two nearest neighbors, so that the elements in each ring touch along a circle. This structure maps into the computational plane as columns of circles, with each column offset from its two nearest neighbors. All of the elements in the computational space are the same size, Fig. 1.7.

This circular case also offers many advantages for simulation, such as its simplicity: it is easy to find equations for the circles in the pixel fitted simulation, and the actual structure is much more like that of the eye than the rectangular structure.

Unfortunately, along with these advantages are also some serious disadvantages. For example, a relatively large area of the sensor is not covered by elements; complete coverage can only be accomplished if neighboring elements can be overlapped. In an actual solid state device this overlap is impossible, so this would be a problem for sensor implementation. In addition to this, the lines separating elements are no longer straight from the

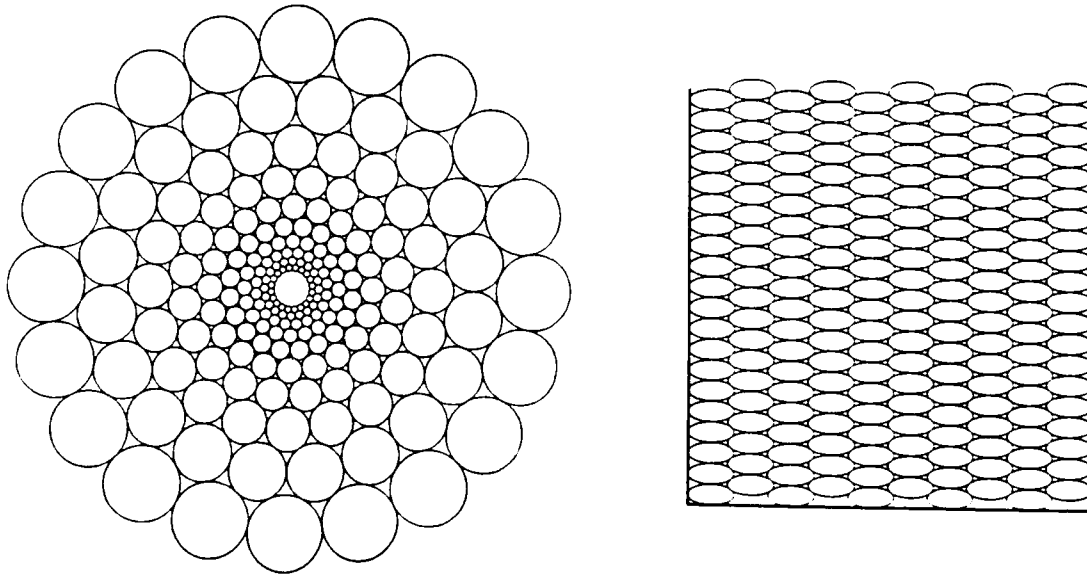


Figure I.7. Image and Computational Planes for Circular Elements.

center to the periphery, which could make line routing more difficult. Since the image plane maps to a circular/columnar structure, the computational plane is neither a rectangular grid, nor a hexagonal one. This means that existing image processing techniques no longer apply to the computational plane. The rectangular array, which is an eight neighborhood, is now replaced by the staggered circular array, which is a special case of the six neighborhood. Even though this structure forces us to develop new processing techniques, the use of a six neighborhood could significantly decrease processing time due to the smaller number of operations, as indicated before.

The final major disadvantage of the circular structure is that the number of elements in each ring determines the inter-ring spacing. The circular elements fit together in such a way that when the number of elements in a ring is specified, the exponential spacing between the rings is set. This means that outside of the specification of the number of elements in each ring, there is no control over the resolution. It also means that when a small number of elements is used, the spacing between the rings increases, which could lead to problems due to the vast differences in area between inner and outer elements.

The geometry of the circular element arrangement was analyzed in depth to determine just how the number of elements in each ring is related to the size of the elements and to the inter-ring spacing. This was done by studying the geometric relations between adjacent elements. By constructing several triangles with vertices at the center of adjacent circles, a mathematical relationship between the elements can be determined, Fig. I.8. In this case, N is the specified number of elements per ring, and the following relations hold.

$$\alpha = \pi/N \quad (1)$$

$$\theta = \pi/2 - \beta/2 \quad (2)$$

$$\delta = \pi/2 - \alpha \quad (3)$$

$$\sin(\gamma/2) = \frac{r_2}{r_1 + r_2} \quad (4)$$

$$\sin(\beta/2) = \frac{r_1}{r_1 + r_2} \quad (5)$$

$$\pi = \gamma/2 + \theta + \delta \quad (6)$$

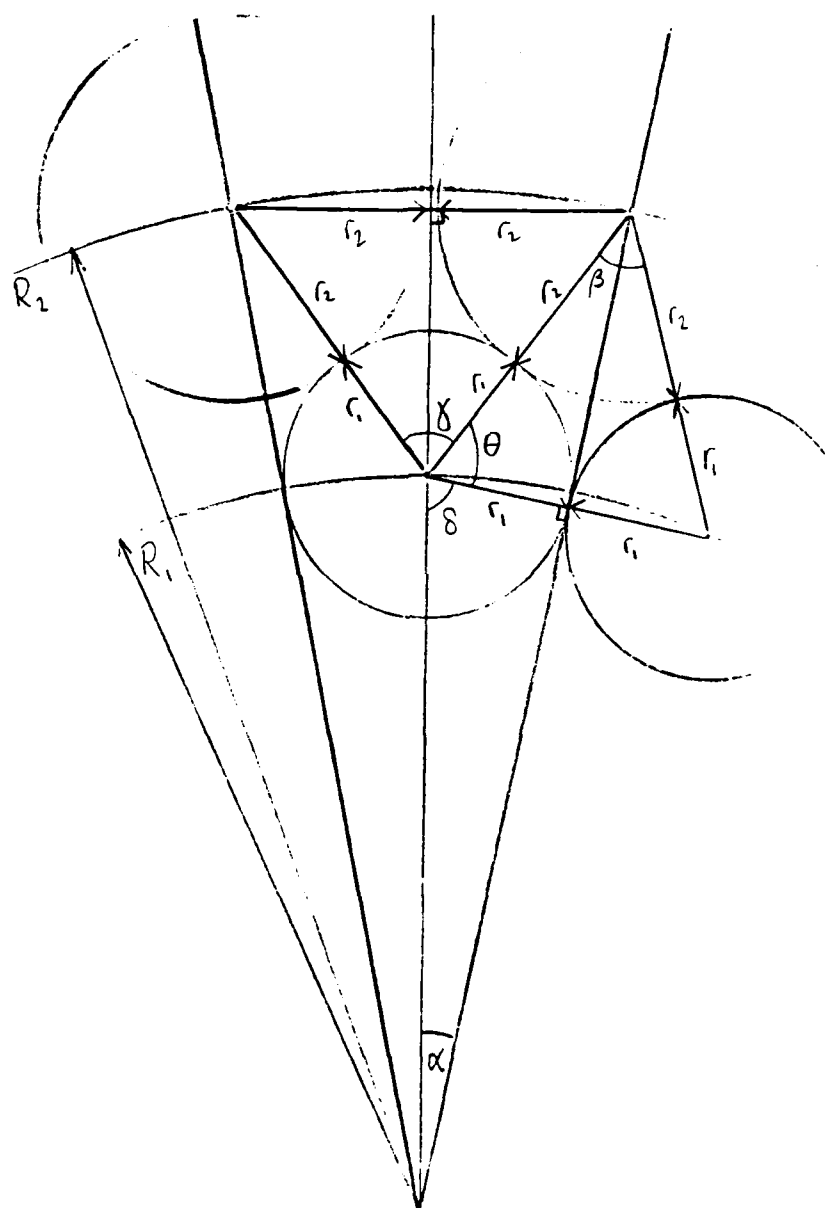


Figure I.8. Relationship Between Adjacent Circular Elements.

By substituting (2) and (3) into (6) and then simplifying, equation (7) is obtained,

$$\alpha = \gamma/2 - \beta/2 \quad (7)$$

From (4),

$$r_2 = (r_1 + r_2) \sin(\gamma/2) \quad (8)$$

$$\frac{r_2}{r_1} = \frac{\sin(\gamma/2)}{1 - \sin(\gamma/2)} \quad (9)$$

From (5),

$$r_1 = (r_1 + r_2) \sin(\beta/2) \quad (10)$$

$$r_1 + r_2 = \frac{r_1}{\sin(\beta/2)} \quad (11)$$

Combining (4) and (11) yields,

$$\frac{r_2}{r_1} = \frac{\sin(\gamma/2)}{\sin(\beta/2)} \quad (12)$$

From (9) and (12),

$$1 - \sin(\gamma/2) = \sin(\beta/2) \quad (13)$$

By further manipulation, it is simple to arrive at an expression for α in terms of γ .

$$\beta/2 = \sin^{-1}(1 - \sin(\gamma/2)) \quad (14)$$

$$\alpha = \gamma/2 - \sin^{-1}(1 - \sin(\gamma/2)) \quad (15)$$

Unfortunately, this equation is transcendental, so it is impossible to obtain a closed form solution. This made it necessary to use a root finding program to solve for γ . Once γ has been determined for a particular α , the other angles and radii follow easily.

$$\frac{r_2}{r_1} = \frac{R_2}{R_1} \quad (16)$$

By normalizing (the sizes will later be scaled to fit the rectangular array) R_1 to 1, R_2 follows immediately from (9), and r_1 and r_2 are similarly simple to find.

$$r_1 = \sin(\pi/N) \quad (17)$$

$$r_2 = R_2 \sin(\pi/N) \quad (18)$$

From this particular case it is quite easy to find the general relationship for the n^{th} ring.

$$R_n = R_2^n \quad (19)$$

$$r_n = R_2^n \sin(\pi/N) \quad (20)$$

From equations (15) through (20) we obtain all of the information necessary to produce the circular simulation in the image plane. The mapping into the computational plane is actually quite simple to simulate, being based on a relatively uncomplicated geometry, Fig. I.9.

The equations to fit this mapping onto a rectangular array are also straightforward. In the equations below, n is the specified number of elements per ring and m is the number of rings.

$$d_h = 2r + (m-1)r_1 \quad (21)$$

$$r_1 = \sqrt{(2r)^2 - r^2} = \sqrt{4r^2 - r^2} = \sqrt{3}r \quad (22)$$

$$d_h = 2r + (m-1)\sqrt{3}r = 2 + \sqrt{3}(m-1)r \quad (23)$$

$$d_v = n(2r) + r = (2n+1)r \quad (24)$$

Now the computational plane situation is just a matter of solving equations (23) and (24) for the r that best fits the computational plane information to an array for display. This step is unnecessary in reality, because

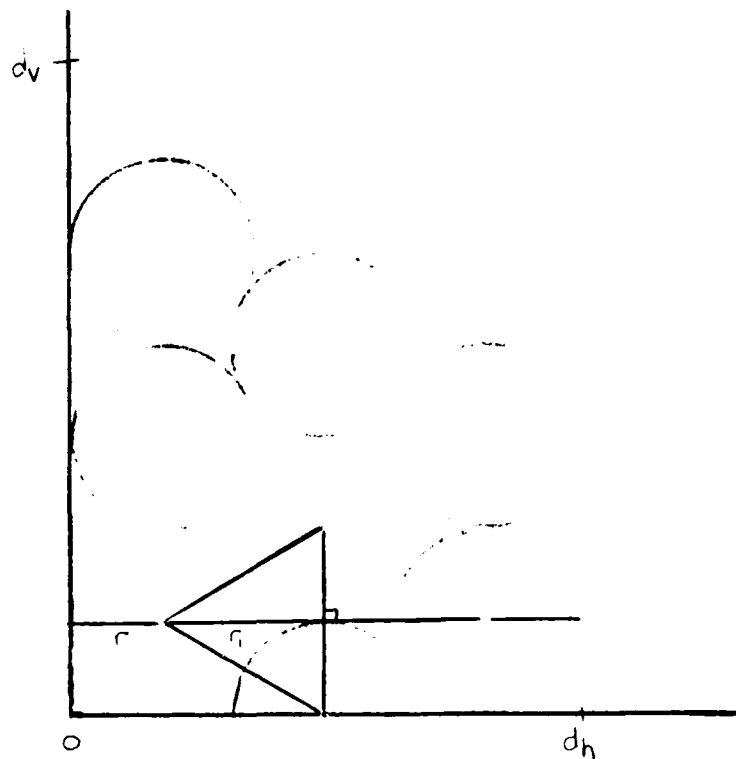


Figure I.9. Computational Plane Relationship for Circular Elements.

only the information is important in the actual sensor. This completes the necessary equations for the circular simulations.

The hexagonal element case is very similar to the circular elements in that it is essentially just an expansion of each circular element until total coverage is obtained. That is, if each circular element is made larger so that all of the sensor area is covered, the intersections of the elements form, very nearly, hexagons. While these hexagons are close to being regular (all sides and angles equal), they are slightly "warped" due to the structure of the sensor. The basic structure is that of concentric rings of increasing radius,

each with the same number of hexagonal elements, with each subsequent ring staggered from its two nearest neighbors. This maps into the computational plane as columns of hexagonal elements, with each column staggered from its two nearest neighbors, Fig. I.10. This design also has advantages and disadvantages, which will be discussed below.

One of the major advantages is, precisely, that all of the sensor area is covered by elements, and the log spiral structure is maintained. This is basically a combination of the good aspects of the other two designs, but there are also bad aspects of this structure.

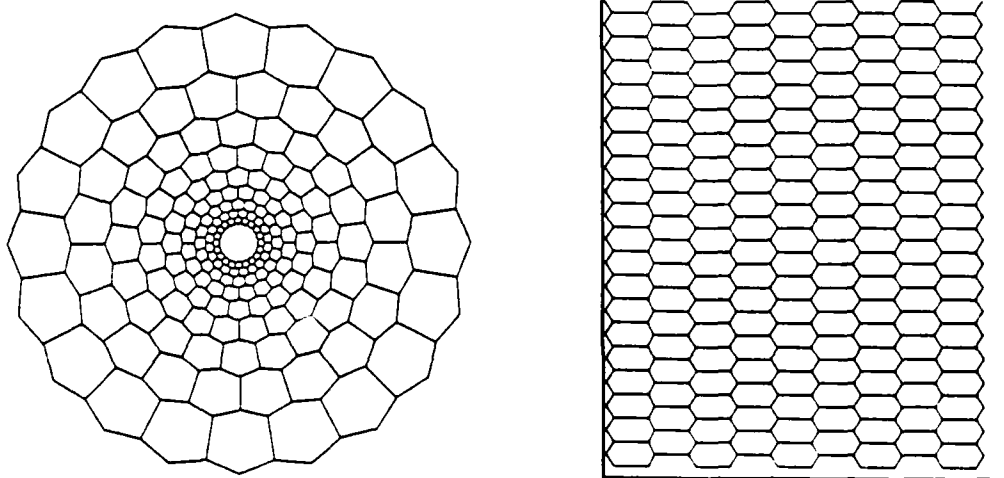


Figure I.10. Image and Computational Planes for Hexagonal Elements.

Most of the disadvantages of this arrangement are the same as those for the circular element case, such as the inability to control inter-ring spacing, and the lack of straight lines from the center to the periphery. In addition to these problems, the hexagons are by far the hardest to simulate (this is a secondary problem with little relevance to the implementation). The rectangular elements are defined by two circles and two lines through the center of the array, which is a fairly simple structure to specify. The circular elements are even easier to define, by using only one circle for each. The hexagons, on the other hand, are composed of six lines, only two of which go through the center of the array. This geometry, which eliminates uncovered sensor area, also complicates simulation.

Since we know that these hexagons must be "warped" slightly to properly fit the structure, it is necessary to determine just how this occurs. To define the geometry of the sensor, the preservation of the angles of the hexagons (see Fig. I.11.) was first investigated. By drawing a model of one element of the array, equations can be derived to express the changes in the angles as a function of the number of elements per ring.

$$\alpha = \pi / N \quad (25)$$

$$\alpha_1 + 2\alpha_4 = 2\pi \quad (26)$$

$$\alpha_2 + 2\alpha_3 = 2\pi \quad (27)$$

$$\alpha_3 + \alpha = \alpha_4 \quad (28)$$

Unfortunately, since N is the given number of elements per ring, there are not enough equations to solve this problem for all of the angles. To get around this, the condition that opposite angles must add to 240 degrees was

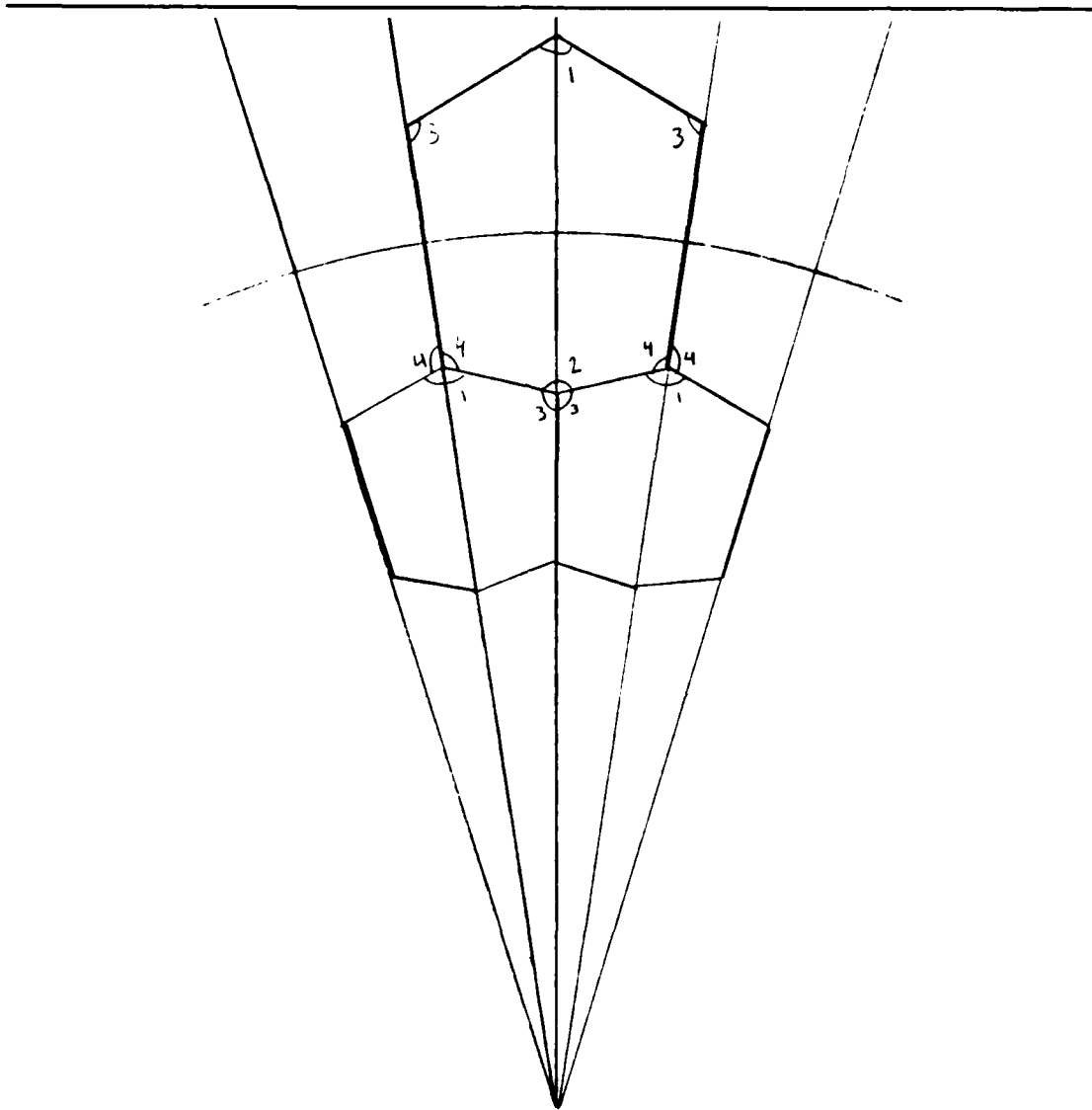


Figure I.11. Angle Relationships for Hexagonal Elements.

imposed (which preserves as much of the regular hexagon's features as possible). This gives two more equations so that the system may be solved.

$$\alpha_1 + \alpha_2 = 4\pi/3 \quad (29)$$

$$\alpha_3 + \alpha_4 = 4\pi/3 \quad (30)$$

Substitution yields the following relationships, which fully characterize the angles in the hexagonal case.

$$\alpha_3 = (4\pi/3 - \pi/N)/2 \quad (31)$$

$$\alpha_4 = 4\pi/3 - \alpha_3 \quad (32)$$

$$\alpha_1 = 2\pi - 2\alpha_4 \quad (33)$$

$$\alpha_2 = 2\pi - 2\alpha_3 \quad (34)$$

Now that the angles are specified, the relationships between the sides can be explored. This is accomplished by creating two elements and making some simple observations about the geometry, Fig. I.12. The rest, even though fairly involved, is just simple algebra.

$$x_3 = y_1 \quad (35)$$

$$\frac{x_2}{x_1} = \frac{x_3}{x_2} = \frac{y_2}{y_1} = \frac{y_3}{y_2} \quad (36)$$

$$a = (r_1 - x_2/2)\sin(\alpha) \quad (37)$$

$$a = x_1\sin(\alpha_2/2) \quad (38)$$

From (37) and (38),

$$x_1 = (r_1 - x_2/2) \frac{\sin(\alpha)}{\sin(\alpha_2/2)} \quad (39)$$

and

$$b = (r_1 + x_2/2)\sin(\alpha) \quad (40)$$

$$b = x_3\sin(\alpha_1/2) \quad (41)$$

From (40) and (41),

$$x_3 = (r_1 + x_2/2) \frac{\sin(\alpha)}{\sin(\alpha_1/2)} \quad (42)$$

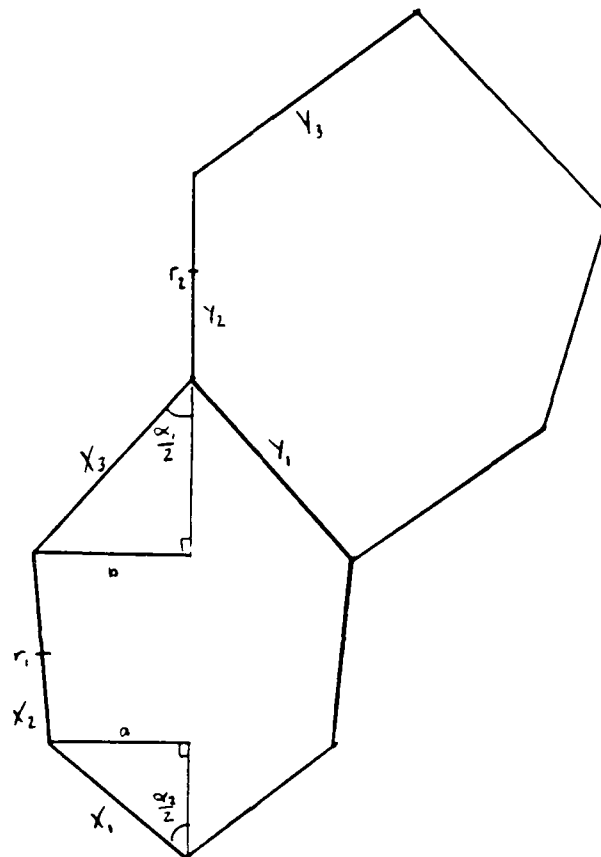


Figure I.12. Relationship Between Adjacent Hexagonal Elements.

At this point, the assumption that the sides were related by the geometric mean, that is $x_1 x_3 = x_2^2$, was made. Thus, by substituting and simplifying, we arrive at the following equation.

$$x_2 = \frac{r_1}{K} \quad (43)$$

where

$$K = \left[\frac{\sin(\alpha_2/2)\sin(\alpha_1/2)}{\sin^2(\alpha)} + 1/4 \right]^{1/2} \quad (44)$$

From equations (43) and (44), several more relationships can be determined by substitution into (39) and (42) shown previously.

$$x_1 = K_1 r_1 \quad (45)$$

$$K_1 = (1 - 1/2K) \frac{\sin(\alpha)}{\sin(\alpha_2/2)} \quad (46)$$

$$x_3 = K_2 r_1 \quad (47)$$

$$K_2 = (1 + 1/2K) \frac{\sin(\alpha)}{\sin(\alpha_1/2)} \quad (48)$$

By making a similar comparison with the larger hexagon, and from (35), (39), and (42),

$$y_1 = x_3 = (r_1 + x_2/2) \frac{\sin(\alpha)}{\sin(\alpha_1/2)} = (r_2 - y_2/2) \frac{\sin(\alpha)}{\sin(\alpha_2/2)} \quad (49)$$

From (43),

$$y_2 = r_2 / K \quad (50)$$

From (43), (49), and (50),

$$\frac{r_1(1 + 1/2K)}{\sin(\alpha_1/2)} = \frac{r_2(1 - 1/2K)}{\sin(\alpha_2/2)} \quad (51)$$

Which reduces to

$$\frac{r_2}{r_1} = \frac{(1 + 1/2K)\sin(\alpha_2/2)}{(1 - 1/2K)\sin(\alpha_1/2)} \quad (52)$$

Now, since K is completely specified (α is a function of N and α_1 and α_2 are determined by the previously described method), the inter-ring spacing constant is found by normalizing r_1 to 1. The general case now follows.

$$r_n = r_0 \left(\frac{r_2}{r_1} \right)^n \quad (53)$$

Where r_0 is the smallest radius.

Using (43) through (53) provides the lengths for all of the sides. This completely specifies all of the geometric information for the hexagonal element case.

Using the mathematical relationships deduced above, the simulations can now be performed.

I.4. Simulations

Based on the equations of Sect. I.3, Computer aided design and simulation were performed. Two types of computer models were developed, the image and computational plane design or structural design, and the pixel fitting or perceptual simulation. In the structural design, a plotting package was used to draw the actual arrangement of the elements according to parameters determined beforehand. In the perceptual simulation, a regular square array camera was used to take images, which were then used to produce pictures simulating the appearance of that image in the image and computational planes of the BVS sensor.

The structural designs were fairly straightforward, once the geometry discussed in the preceeding section was worked out. They essentially consisted of using the equations found and drawing the boundaries of the elements in both the image and computational planes.

The rectangular case was first explored by writing a program to, from set parameters, calculate all of the necessary equations and create a file that is readable by the plotter package. This program creates both the image and

computational plane structures. The program is shown in Appendix A, and the type of structure obtained can be seen in Appendix B, Figures B1 and B2 (image and computational planes respectively), or in the text as Figures I.5. and I.6.

The circular case was designed in a similar fashion, except that it was necessary to create a root finding subroutine to solve the inter-ring spacing equation. Once this is done, the program calculates all of the proper radii and creates the image and computational plane plot files. This program is also in Appendix A and examples of these designs are found as Figure I.7. in the text and in Appendix B, Figures B3 and B4 (again image and computational planes).

The most difficult of the three structural simulations was the hexagonal element case. The difficulty was due to the number of lines used to specify each hexagon. The program calculates all of the locations for the vertices from given information, then creates the files to be plotted. The program and examples are located, as in the other two cases, in Appendix A and Appendix B, Figures B5 and B6, respectively, as well as in the text as Figure I.10.

All of these simulations produced good results, and the logarithmic structure can be seen easily in the circular and hexagonal cases. It is also apparent that the difference in size between inner and outer elements in the circular and hexagonal situations could be a problem for implementation. The number of elements used in all cases is small, but should prove to be sufficient in some applications. The actual number of elements will depend,

of course, on the required resolution.

The second type of simulation, the perceptual simulation, was done using the EE Department image processing equipment to mimic the appearance of images in the image and computational planes. Images were created using a 512x512 CCD camera and support equipment. The programs read in these images, then, according to the calculated geometry, pixel-fits the image to the new structure.

The process works as follows: the geometry of each element is specified (location and boundary), then the image is examined until all of the pixels of that image that are inside the specified boundary are found, then these pixels' intensity values are averaged[†] and all of the area covered by these pixels is set to the average value. This is done for each element, and the information is saved to be mapped into the computational plane.

The computational plane processing is done similarly, with the element first being located and its boundary found. Then the value of the element from the image plane that corresponds to this computational plane element is used as the value for all of the pixels inside the boundary. This process is repeated for each element until the mapping is complete. This simulation could become much more complicated if hardware preprocessing is inserted between the image and computational planes.

The programs then write out both the image plane image and the computational plane image in a form suitable for display.

[†] This is done in order to get an equivalent average intensity for the element.

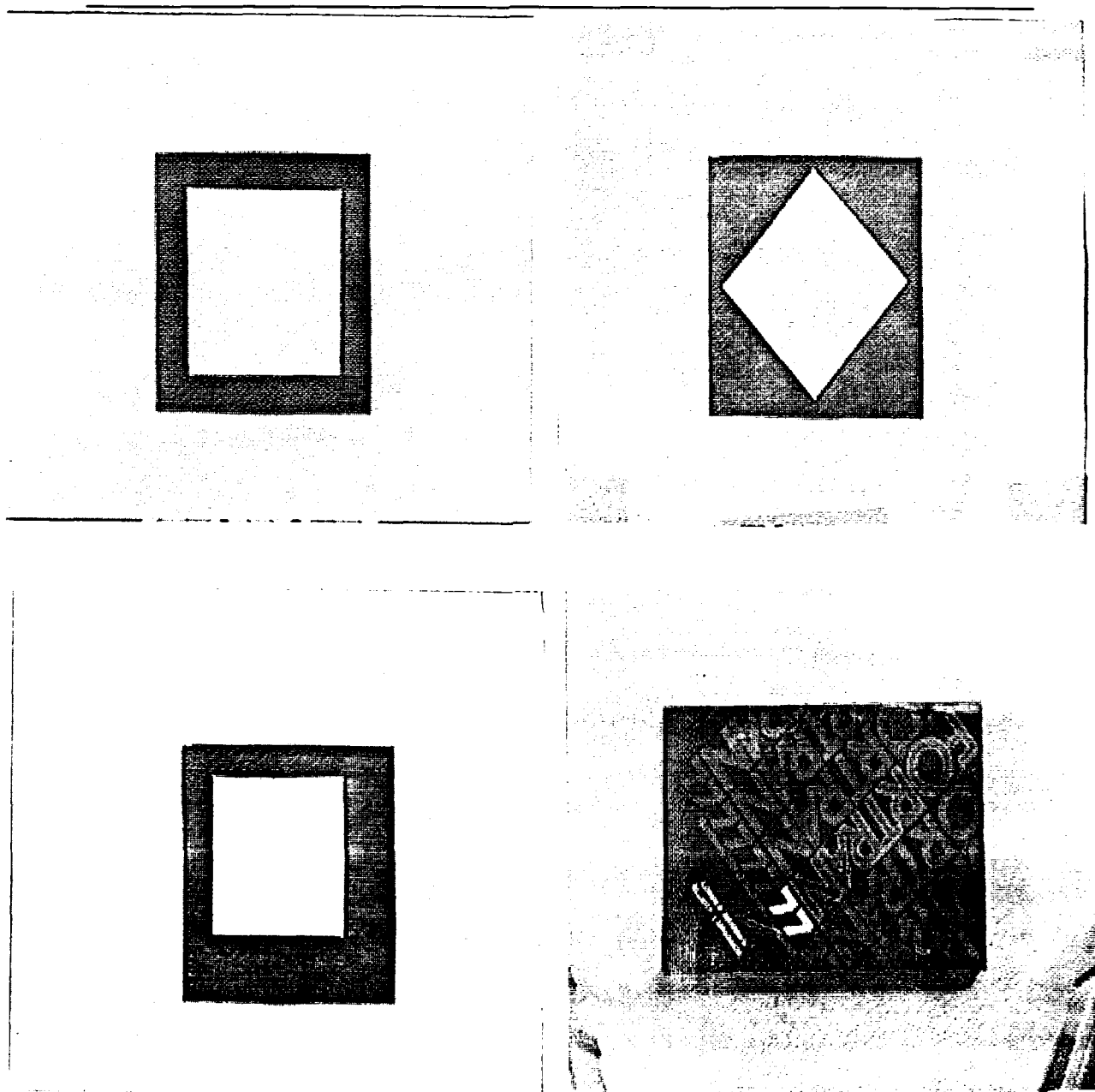


Figure I.13. Original Images.

The four original images are shown in Fig. I.13.

The rectangular element case was the simplest to simulate, and the results are very encouraging. The resolution in the image plane is good, as shown in Figs. I.14(a) to I.14(d) even though there is a relatively small number of elements being used. Results show that usable information can be obtained from a small number of elements with a drastic savings in total data to be processed (3840 elements are used in Fig. I.14.). The computational plane situation gives us almost exactly what we expect from theory, and the mapping does have the important form invariance characteristics (Form invariance under magnification is shown in Fig. I.14(a), and form invariance under rotation is shown as Fig. I.14(b).). The distortion due to shifting was also quite evident, Fig. I.14(c).

It is also evident that the ability to control the inter-ring spacing is a definite advantage. A great increase in resolution was obtained just by decreasing the inter-ring spacing. This made the size change between rings smaller, and allowed more elements to be used; hence, better resolution. Unfortunately, this cannot be done directly in the other two cases.

The rectangular and circular simulation programs are shown in Appendix A.

The circular simulation was also performed with favorable results. The image and computational planes give images that are almost completely as expected. The image plane simulation clearly shows where the sensor is not covered by elements, and the nonrectangular computational plane arrangement is also evident, Figs. I.15(a) to I.15(d). Again, form invariance under

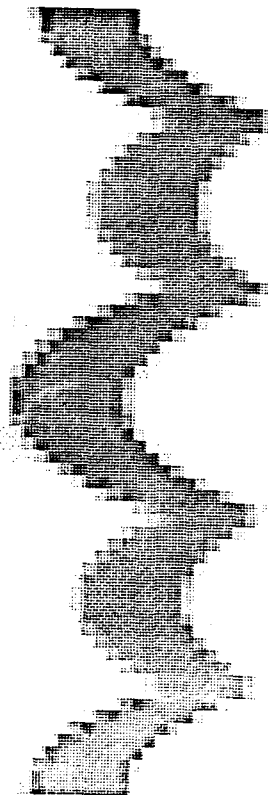
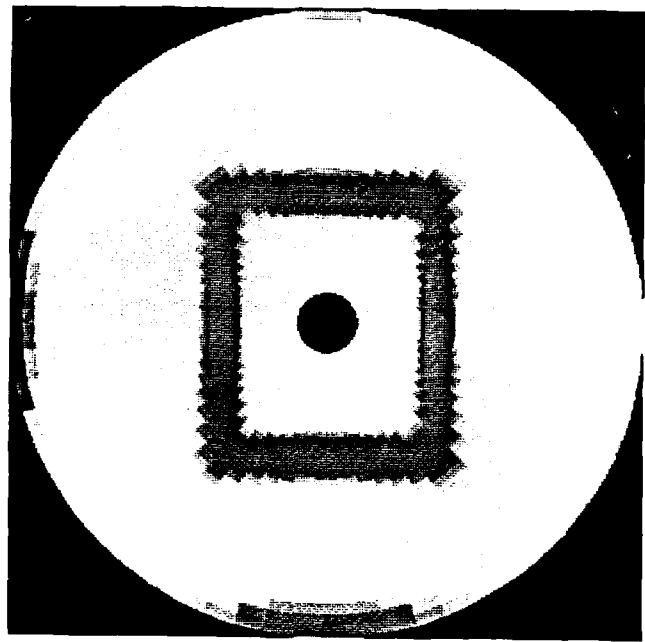


Figure I.14(a). Rectangular Element Simulation of (1) in Fig. I.13.
Demonstrates Form Invariance Under Magnification.

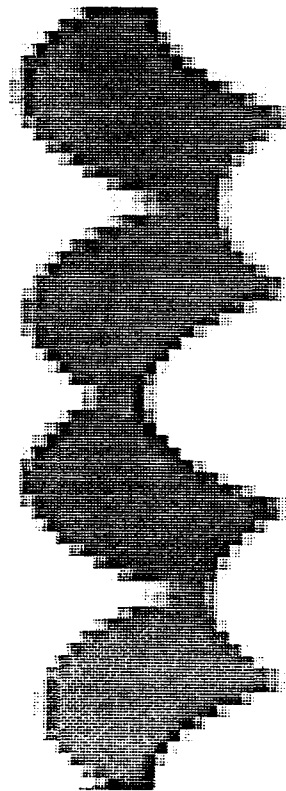
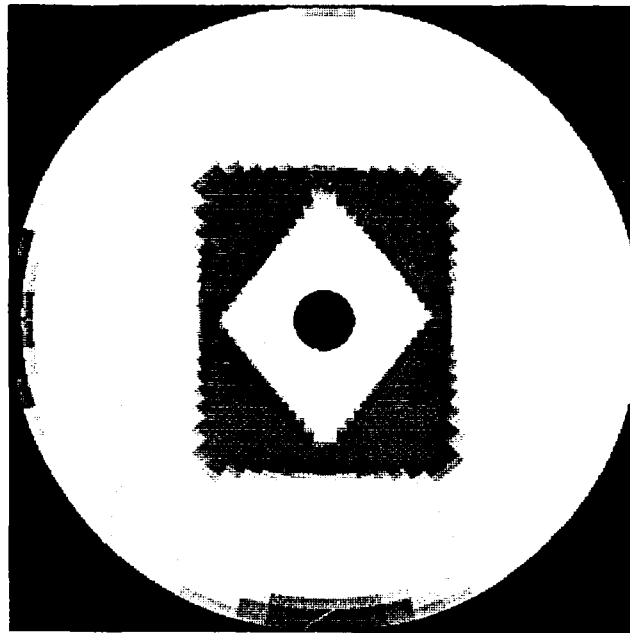


Figure I.14(b). Rectangular Element Simulation of (2) in Fig. I.13.
Demonstrates Form Invariance Under Rotation.

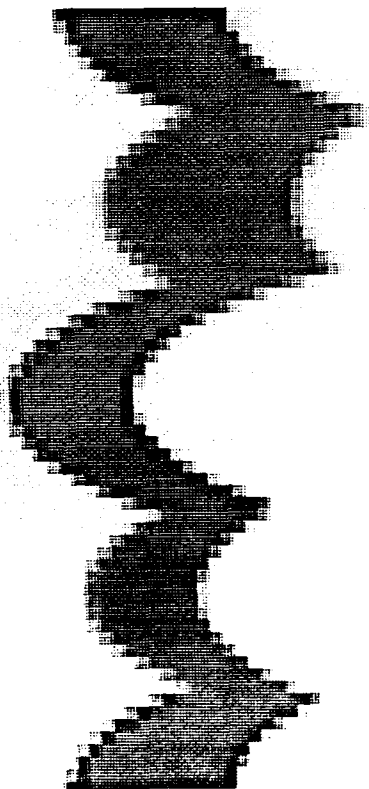
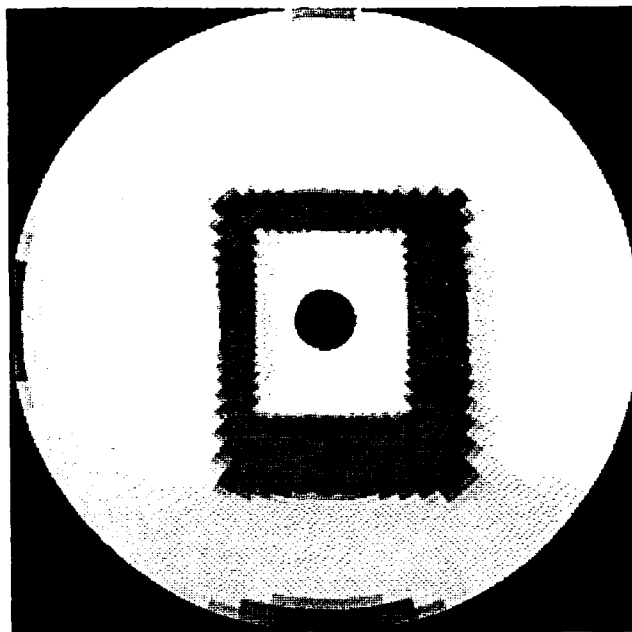


Figure I.14(c). Rectangular Element Simulation of (3) in Fig. I.13.
Demonstrates Distortion Due to Shifting.

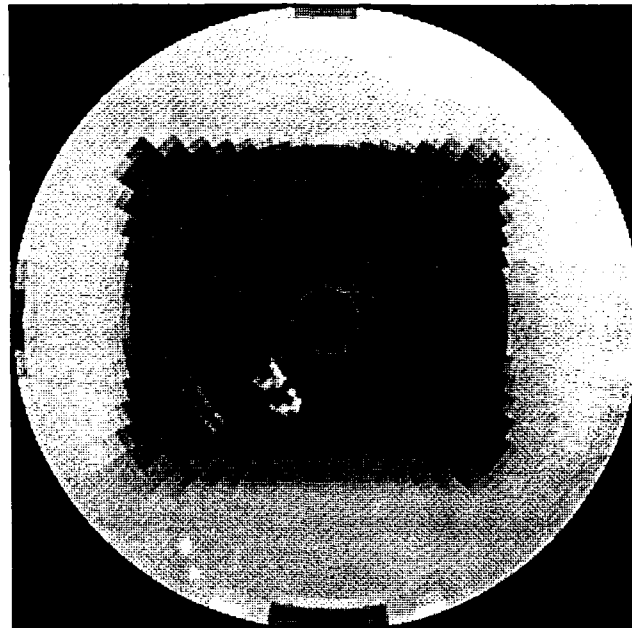


Figure I.14(d). Rectangular Element Simulation of (4) in Fig. I.13.
Demonstrates Resolution of Simulation.

magnification and rotation are demonstrated, Figs. I.15(a) and I.15(b) respectively, and the shift induced distortion is fairly obvious, Fig. I.15(c).

It is also easy to see that the inability to directly change the inter-ring spacing may limit the flexibility of this arrangement. Even though this spacing cannot be controlled, the resolution is still quite good, in both planes, Fig. I.15(d).

The hexagonal structure simulation was not finished to be included in this report.

I.5. Verification of Properties

This section is concerned with the verification of the form invariance under magnification and rotation properties of the complex logarithmic spiral conformal mapping (LSM), for both the rectangular and circular element simulations.

The rectangular simulation did indeed demonstrate the property of form invariance under magnification. This is shown in Fig. I.14(a). In the case shown, two squares are shown one inside the other—this is the same as a magnification of a square (the "squares" appear as rectangles due to the distortion caused by the camera's rectangular elements). The mapping gives us exactly the same shape for both of the squares, verifying the property indicated by theory.

The circular simulation provides similar verification of this property when the same original image is used. This is shown in Fig. I.15(a). As in the rectangular case, the mapping gives the same edge for both cases, with

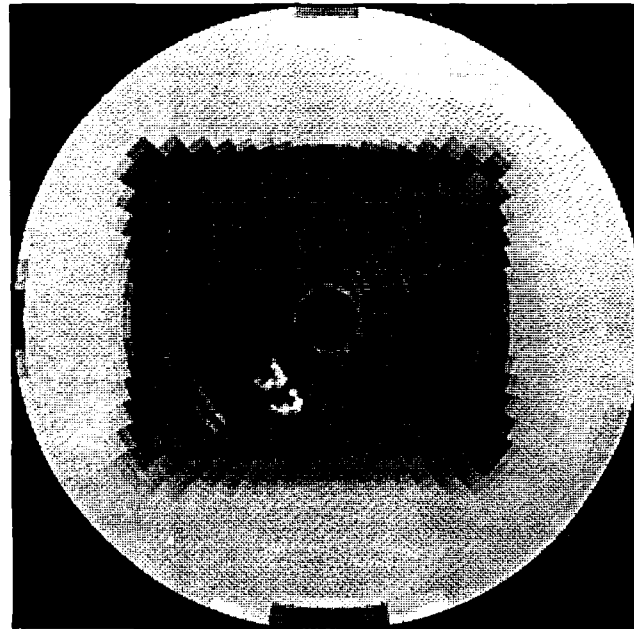


Figure I.14(d). Rectangular Element Simulation of (4) in Fig. I.13.
Demonstrates Resolution of Simulation.

magnification and rotation are demonstrated, Figs. I.15(a) and I.15(b) respectively, and the shift induced distortion is fairly obvious, Fig. I.15(c).

It is also easy to see that the inability to directly change the inter-ring spacing may limit the flexibility of this arrangement. Even though this spacing cannot be controlled, the resolution is still quite good, in both planes, Fig. I.15(d).

The hexagonal structure simulation was not finished to be included in this report.

I.5. Verification of Properties

This section is concerned with the verification of the form invariance under magnification and rotation properties of the complex logarithmic spiral conformal mapping (LSM), for both the rectangular and circular element simulations.

The rectangular simulation did indeed demonstrate the property of form invariance under magnification. This is shown in Fig. I.14(a). In the case shown, two squares are shown one inside the other—this is the same as a magnification of a square (the "squares" appear as rectangles due to the distortion caused by the camera's rectangular elements). The mapping gives us exactly the same shape for both of the squares, verifying the property indicated by theory.

The circular simulation provides similar verification of this property when the same original image is used. This is shown in Fig. I.15(a). As in the rectangular case, the mapping gives the same edge for both cases, with

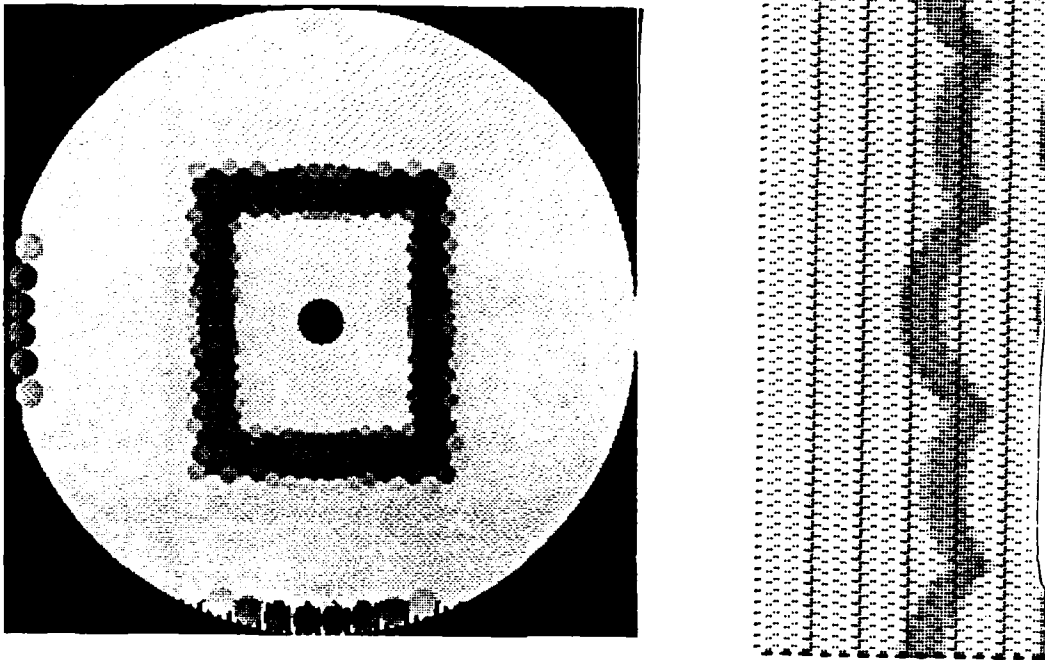


Figure I.15(a). Circular Element Simulation of (1) in Fig. I.13.
Demonstrates Form Invariance Under Magnification.

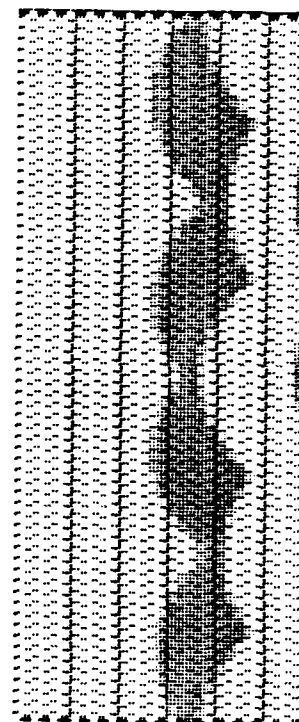
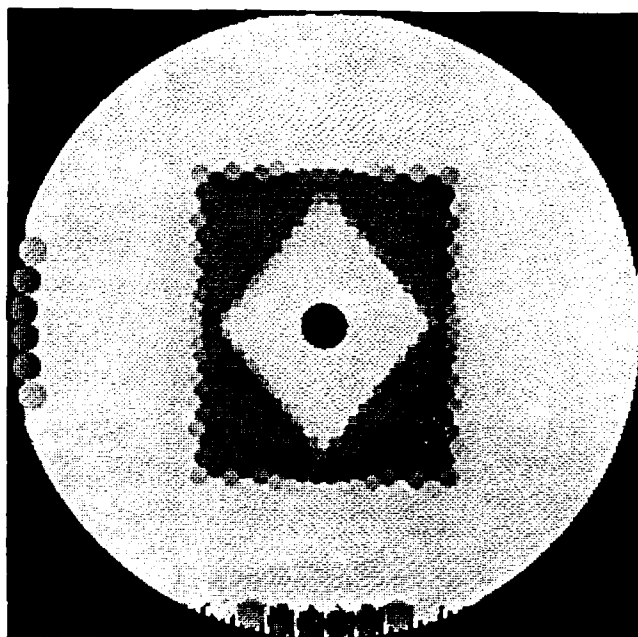


Figure I.15(b). Circular Element Simulation of (2) in Fig. I.13.
Demonstrates Form Invariance Under Rotation.

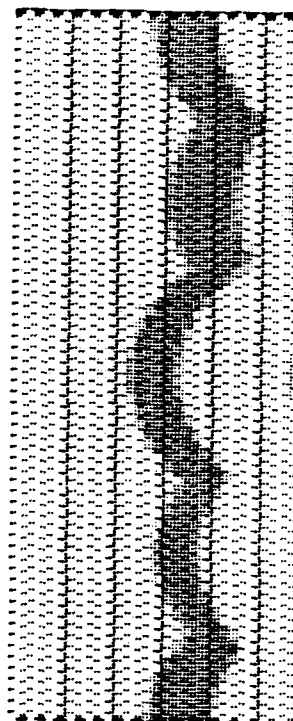
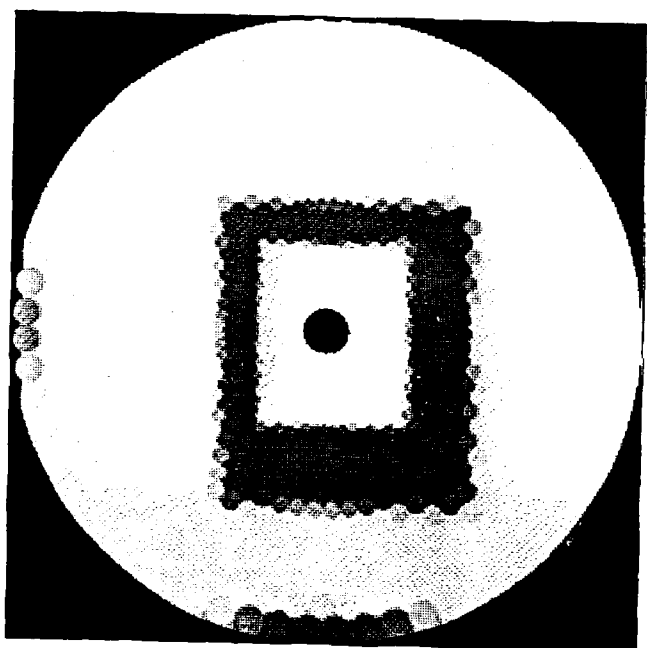


Figure I.15(c). Circular Element Simulation of (3) in Fig. I.13.
Demonstrates Distortion Due to Shifting.

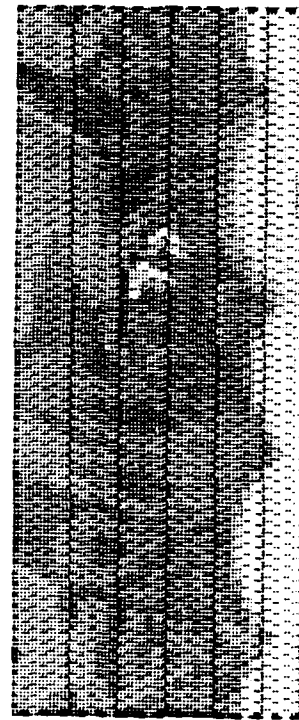
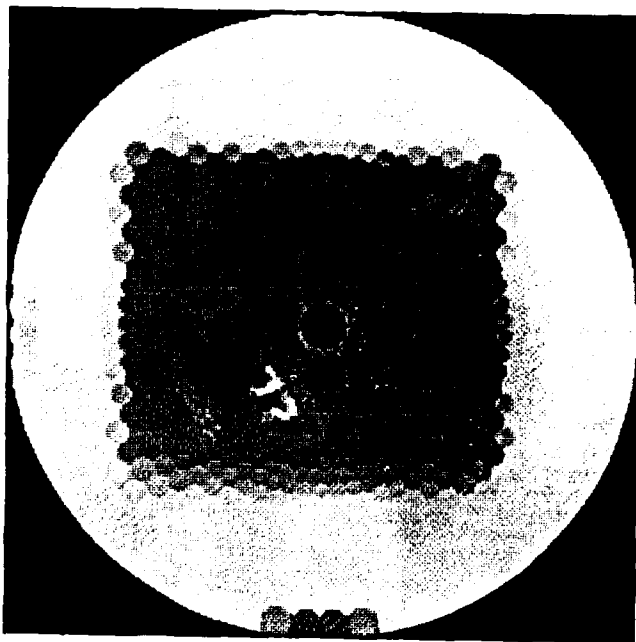


Figure I.15(d). Circular Element Simulation of (4) in Fig. I.13.
Demonstrates Resolution of Simulation.

a shifting in the x direction only, providing further proof of the validity of the form invariance under magnification.

The rectangular simulation was performed on the rotated (and magnified) superimposed squares in an effort to verify the form invariance under rotation. This simulation did demonstrate that the theory was correct and that the shape in the computational plane under the mapping is the same for both of the rotated squares. This is shown in Fig. I.14(b).

Similar results were obtained using the circular simulation on the same original image. Again the property was verified and the demonstration appears in Fig. I.15(b). The shape is shifted only in the y direction, but is identical to the nonrotated shape.

Information is lost due to the nonoverlapping nature of the circular simulation. Since the elements are circular, they cannot cover all of the area when placed in this structure, so some of the information is lost.

This is shown in Figs. B3 and B4 in appendix B and Figs. I.16 and I.17. The space that is not covered is clearly visible in all cases, thus this particular aspect of the theory is also obtained in the simulation.

I.6. Comparison to the Conventional Sensor

This section deals with the direct comparison of the simulations to the conventional sensor (i.e. a rectangular array comprised of rectangular pixels) for both the rectangular and circular element cases. The three areas of comparison are resolution, number of elements, and field of view. We would expect to have comparable resolution near the fovea in our design, a great

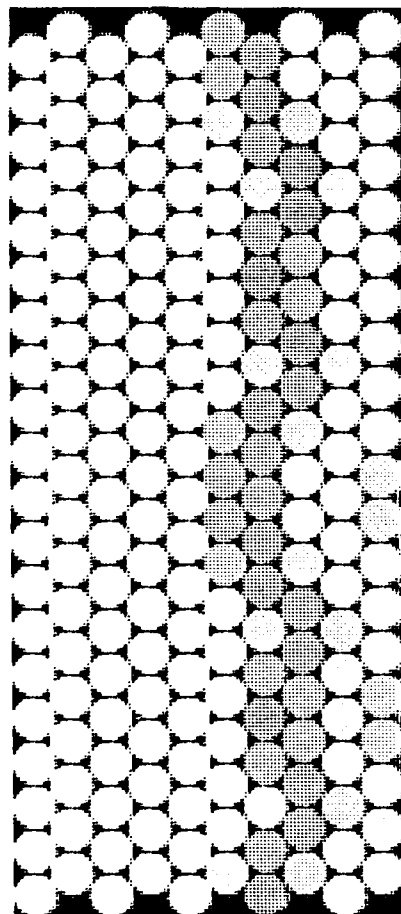


Figure I.16. Computational Plane of Circular Element Simulation.
Demonstrates Uncovered Sensor Area.

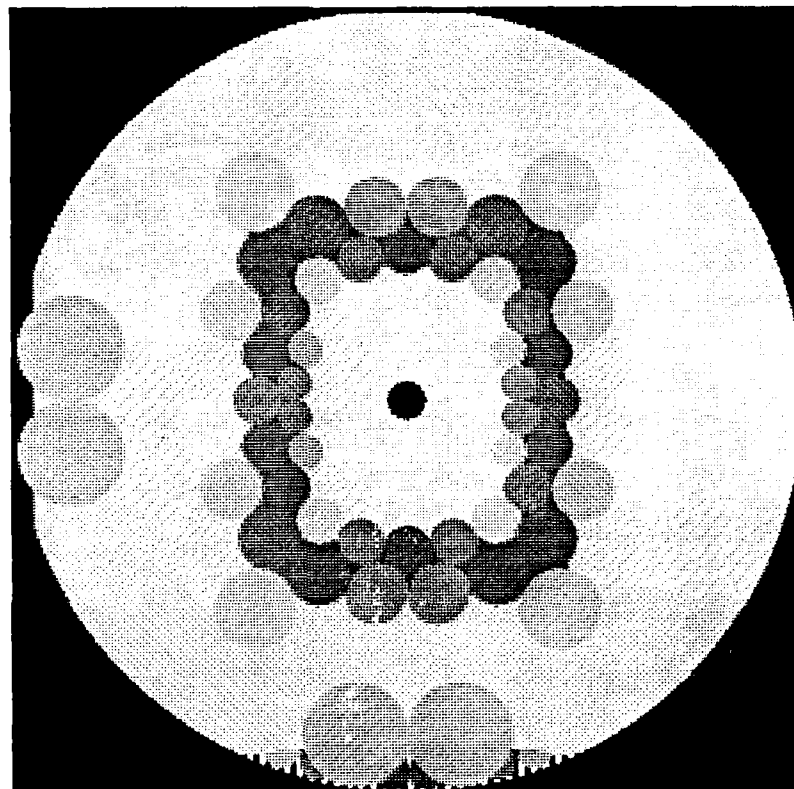


Figure I.17. Image Plane for Circular Element Simulation.
Demonstrates Uncovered Sensor Area.

savings in the number of elements, and a greatly increased field of view.

I.6.1. Resolution

The comparison of resolution in our current simulations was performed geometrically, based on the actual number of elements used. The drastic savings was evident in both the rectangular and circular cases, with both the fovea blank and with the fovea filled with normal rectangular elements.

The current rectangular simulation uses 64 elements per ring for 60 rings, or a total of 3840 elements. The area of the smallest rectangular element is roughly 2.36 pixels, so the resolution at the fovea is about half that of the normal sensor. This calculation is very rough and should be repeated with better precision (the biggest error comes from the assumption that the pixels are square, instead of their actual rectangular shape). Still, this resolution is equivalent to that of a 362x362 array, which is quite good in many applications.

The current circular simulation uses 64 elements per ring in 35 rings, for a total of 2240 elements. The smallest element's area is about 1.35 pixels (again this is a rough calculation), which makes our resolution fairly comparable to the normal array of 512x512 pixels. This resolution near the fovea is very good and useful in many situations.

The real drastic differences should come in when these resolutions are observed in light of the greatly decreased number of elements.

1.6.2. Number of Elements

The number of elements in our array was compared to that of the normal sensor for the rectangular and circular simulations under two conditions: without fovea filling and with fovea filling. This experiment should give us an idea of the savings that this sensor may provide.

The rectangular element case, at 64x60, gave us 3840 elements. Comparing to our 512x512 array, we are now using only 1.46% the number of elements that are used in the normal sensor, which gives us an amazing

computational savings. If the fovea is included, we now have 5700 elements total, which is only 2.17% of the normal sensor elements—again a major savings.

This was based on a 512x512 array, but we have shown that our resolution is only half that of the 512x512 array. If we compare to the 362x362 array that has roughly the same resolution as ours, we still have only 2.72% with no fovea and 4.34% with the fovea. This means that even at a comparable resolution the computational savings are enormous.

In the circular element case, at 64x35, we have 2240 elements and about 512x512 resolution near the fovea. So the direct comparison to the normal sensor means that we are using only 0.85% of the total number of elements in the normal case! Even if we populate the fovea this gives us only 2702 elements or 1.03%. This clearly demonstrates that the near fovea resolution is more than justified by the huge computational savings, even if the other properties can never be used.

1.6.3. Field of View

This section compares the field of view of our sensor to that of a rectangular sensor with the same number of elements. It covers the cases of rectangular elements, circular elements, fovea open, fovea covered, direct comparison and "normalized" comparison. Since our array is circular and the comparison is made using a square array of square pixels, there are three different field of view measurements— Minimum (minimum distance to square boundary), Maximum (maximum distance to square boundary) and

Average (average distance to square boundary).

In the rectangular case without fovea and making a direct comparison, a normal sensor with 3840 elements would have a field of view of only 14.8% (ave. min.=12.1%, max.=17.2%) of the field of view (FOV) of our sensor. If our fovea is populated, giving 5700 elements, a normal array with 5700 elements has a FOV of only 18.0% (ave. min.=14.8%, max.=20.7%) of our FOV. This shows that even though we are using a vastly smaller number of elements and our near foveal resolution is pretty good, we still have a much greater field of view than a normal sensor with a similar number of elements. These calculations are based on a 512x512 array, but we have shown our resolution to be less than that, so we can "normalize" this to the actual case.

The normalized case uses a square array of square elements, each with area equal to the area of the smallest rectangular element in our sensor. This "normalizes" the FOV calculations. Without the fovea in the rectangular element case, a normal FOV for the "normalized" case would be 20.9% (ave. min.=17.1%, max.=24.3%) of our FOV. With the fovea it becomes 25.5% (ave. min.=20.8%, max.=29.3%) of our FOV. So even after normalization our sensor still has a much better FOV than a comparable normal sensor.

Since the circular elements have area roughly that of the 512x512 case, we do not need to normalize to get an idea of the FOV for these situations. A square array of 2240 pixels (no fovea) has a FOV of 11.3% (ave. min.=9.4%, max.=13.3%) of our sensor's FOV. With the fovea filled (2702 elements), a normal sensor has only 12.5% (ave. min.=10.2%, max.=14.5%)

the FOV that our sensor provides. So again our sensor has a drastically greater FOV than a normal sensor with the same number of elements (with area the same as the smallest element in our array).

I.7. Conclusions

The results to date indicate that this new sensor design will provide benefits over current image detection devices. The simulations demonstrate the properties that were theorized, and the proposed savings appears to be evident. The savings in the number of elements is a dramatic testimony of the kind of computational advantage that we may hope to achieve. The resolution of this sensor is another favorable indication of the possible advantages of this configuration. The improvement in field of view is an area that also justifies the possible use of this sensor.

A large quantity of basic research remains to be completed, including the hexagonal simulation. Once the simulations are complete, they can be used for a variety of tests to determine if the benefits of this sensor are worthwhile enough to explore further.

REFERENCES

- [1] Shultze, M. "Zur Anatomie und Physiologie der Retina," *Arch. Mikrosko. Anat.*, Vol. 2, 1866, pp. 175-286.
- [2] Daniel, P. M. and Whitteridge, D., "The Representation of the Visual Field on the Cerebral Cortex in Monkeys," *J. Physiology*, Vol. 159, 1961, pp. 203-221.
- [3] Sandini, G. and Tagliasco, V., "An Anthropomorphic Retina-like Structure for Scene Analysis," *Computer Graphics and Image Processing*, Vol. 14, 1980, pp. 365-372.
- [4] Lettvin, J., Maturana, H., McCulloch, W., and Pitts, W., "What the Frog's Eye Tells the Frog's Brain," *Proceedings of the I.R.E.*, Nov. 1959, pp. 1940-1951.
- [5] Michael, C., "Retinal Processing of Visual Images," *Scientific American*, May 1969, pp. 104-114.
- [6] Schwartz, E., "Spatial Mapping in the Primate Sensory Projection: Analytic Structure and Relevance to Perception," *Biological Cybernetics*, Vol. 25, 1977, pp. 181-194.
- [7] Weiman, C., and Chaikin, G., "Logarithmic Spiral Grids for Image Processing and Display," *Computer Graphics and Image Processing*, Vol. 11, 1979, pp. 197-226.
- [8] Weiman, C., and Chaikin, G., "Logarithmic Spiral Grids for Image Process-

ing," *Proceedings of the Conference on Pattern Recognition and Image Processing Number 2*, 1979.

[9] Braccini, C., Gambardella, G., Sandini, G., and Tagliasco, V., "Borrowing from the Eyes to Create Robot Vision Algorithms," *Sensor Review*, April 1981, pp. 68-72.

[10] Schwartz, E. L., "Afferent Geometry in the Primate Visual Cortex and the Generation of Neuronal Trigger Features," *Biological Cybernetics*, Vol. 28, 1977, pp. 1-14.

[11] Braccini, C., Gambardella, G., Sandini, G., and Tagliasco, V., "A Model of the Early Stages of the Human Visual System : Functional and Topological Transformations Performed in the Peripheral Visual Field," *Biological Cybernetics*, Vol. 44, 1982, pp. 47-58.

[12] Weiman, C., and Chaikin, G., "Conformal Computational Geometry for Machine Vision," *5th International Conference on Pattern Recognition*, Dec. 1980, pp. 1106-1110.

[13] Jain, R., and O'Brien, N., "Ego-Motion Complex Logarithmic Mapping," *S.P.I.E.*, Nov. 1984.

[14] Messner, R. A., and Szu, H. H., "Coordinate Transformation from an Image Plane Directly to an Invariant Feature Space," *Proceedings of the IEEE CVPR Conference*, 1983, pp. 231-240.

[15] Messner, R. A., and Szu, H. H., "Simultaneous Image Processing and Feature Extraction for Two-Dimensional Non-Uniform Sensors," *Proceedings of the SPIE*, Vol. 449, 1983, pp. 693-710.

- [16] Messner, R. A., and Szu, H. H., "An N2 Parallel Architecture for Real Time Scale/Rotational Invariant Image Processing," to appear in *Computer Vision, Graphics, and Image Processing*, 1985.
- [17] Scholten, D., and Wilson, S., "Chain Coding with a Hexagonal Lattice," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 5, Sept. 1983, pp. 526-533.
- [18] Golay, M., "Hexagonal Parallel Pattern Transformations," *IEEE Transactions on Computers*, Vol. C-18, No. 8, Aug. 1969, pp. 733-740.
- [19] Yajima, S., Goodsell, J., Hiraishi, H., and Ichida, T., "Data Compression for Kanji Character Patterns Digitized on the Hexagonal Mesh," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 2, Mar. 1981, pp. 221-230.
- [20] Burt, P., "Tree and Pyramid Structures for Coding Hexagonally Sampled Binary Images," *Computer Graphics and Image Processing*, Vol. 14, 1980, pp. 271-280.
- [21] Mersereau, R., "The Processing of Hexagonally Sampled 2D Signals," *Proceedings of the IEEE*, Vol. 67, No. 6, Jun. 1979, pp. 930-949.
- [22] Sachs, F., "Imaging Devices and Cameras for Automation Processes," *General Electric Company*. Syracuse, N.Y., pp. 1-9.
- [23] Carbone, J., and Hunter, D., "Use of Charge Injection Device Components in Still Cameras," *General Electric IVSO*, 1983, pp. 1-3.
- [24] Carbone, J., Lunden, J., and Michon, G., "VLSI Solid State Sensors for ENG Cameras," *General Electric IVSO*, pp. 1-6.

CHAPTER II

MODELLING EDGE DETECTION IN THE HUMAN VISUAL SYSTEM AND USING THE NEW SENSOR

Introduction

The aim of this chapter is to present two different simulations of edge detection, one for the human peripheral visual system, by giving a functional interpretation of the retina and the lateral geniculate nucleus, the other for the new sensor.

The simulation of edge detection for the human peripheral visual system will emulate the functions of the retinal ganglion cells and lateral geniculate nucleus which perform preprocessing between the retina and the striate cortex. After this preprocessing, the result is mapped to the striate cortex (or computation plane). The preprocessing of the two - dimensional brightness distribution available at the retina, by the complex logarithmic retinotopic mapping, may possibly play a role in the form invariance property of the image plane - computation plane mapping [1,2].

The simulation of edge detection for the new sensor assumes a direct mapping (i.e. a simple projection) of the image from the retina (image plane) into the striate cortex (computation plane) without any intermediate processing. Thus, the edge detection is implemented directly in the computation plane (striate cortex). Although this is not exactly the processing which the

human peripheral visual system performs, it preserves the most important characteristic - 'form invariance'. Also, it is very convenient for further processing, such as motion detection, using image processing techniques.

The organization of this chapter is as follows: In Sect. II.1 an analysis of the cells and the structure of the peripheral visual system is presented. In Sect. II.2 a simple experiment on the peripheral visual system and some of its properties are discussed. The mathematical formalization and some simulation examples are presented for the human peripheral visual system in Sect. II.3. In Sect. II.4 the simulation for the new sensor is presented. In Sect. II.5 savings in computations for the new sensor compared to rectangular sensor is discussed. In the last section, the objectives of our work are discussed and the two different simulations are compared.

II.1. Properties of the Peripheral Visual System

In this section, some properties of the peripheral stages of the human visual system (HVS), i.e. the retinal ganglion cells and the lateral geniculate nucleus, are reviewed. Fig. II.1(a) represents the HVS. The relationship of the peripheral visual system with the retina and the visual cortex is shown in Fig. II.1(b).

II.1.1. Retinal Ganglion Cells

The image perceived by the cones and rods in the retina is processed by the retinal ganglion cells. In the vertebrate retina the cell bodies of the neurons are arrayed in three distinct layers [3]. The outermost layer (the one farther from the lens of the eye) consists of the receptor cells. The next layer includes the bipolar cells (which conduct messages from the receptors

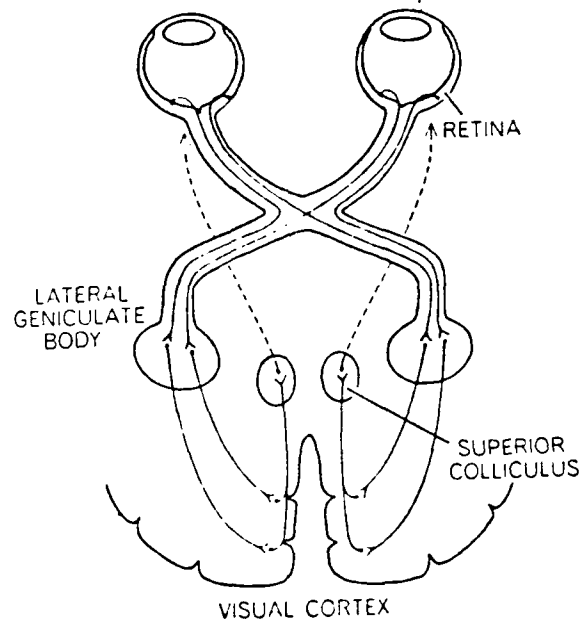


Fig. II.1(a). The Human visual system [3].

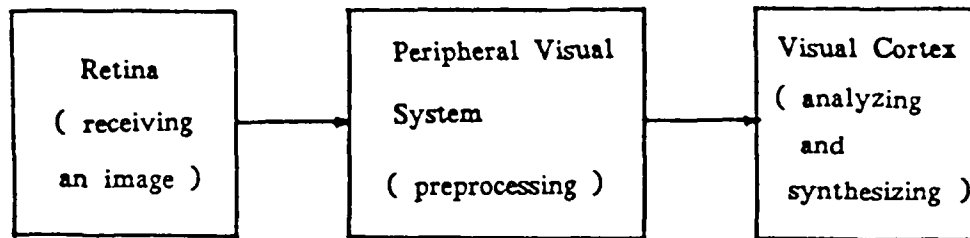


Fig. II.1(b). Relationship of peripheral visual system with retina and visual cortex.

to the cells in the third layer) and the horizontal and amacrine cells, which appear to be involved in the lateral transmission of information. The third layer contains the ganglion cells, whose axons form the optic nerve, the sole

output of the retina. Fig. II.2 shows the different neuron layers in the vertebrate retina.

A given retinal ganglion cell receives information from a rather small population of the receptor cells. The area covered by these receptors is called the receptive field of that ganglion cell. The receptive field of a ganglion cell can be subdivided into two classes, one characterized by an ON center and an OFF surround, the other by an OFF center and an ON surround [4]. Since the responses of these different parts of the receptive field all converge on and share a common path - the axon of the ganglion cell - it is evident that the intensity, the color, and the spatial and temporal configuration of the pattern of illumination falling on receptive field may all be significant factors in determining the ultimate pattern of impulses which the cell

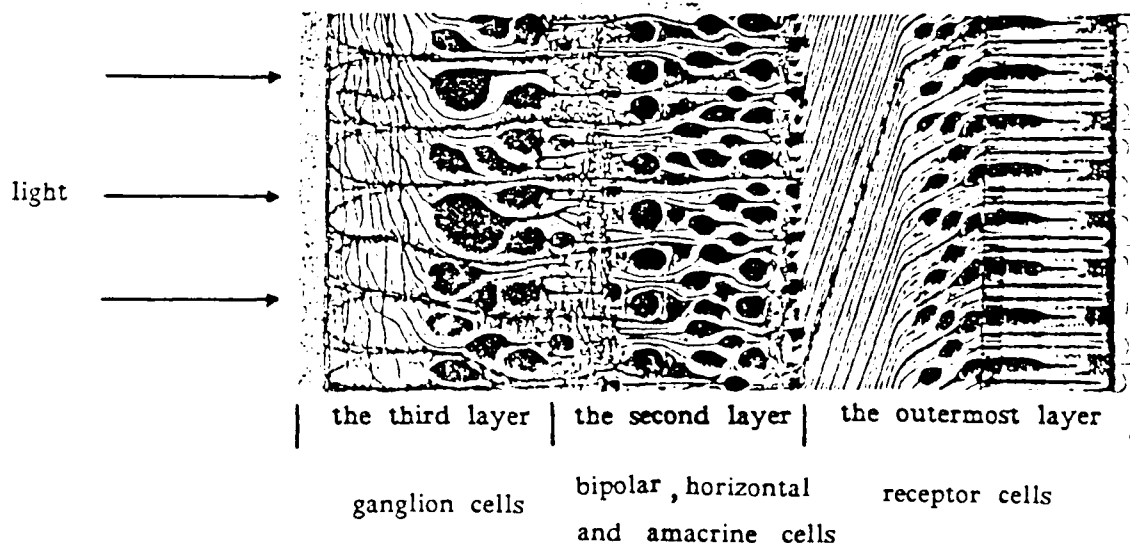


Fig. II.2. The different neuron layers in the vertebrate retina [19].

discharges. Combining these two types of retinal ganglion cells produces neural responses with maximum and minimum adjacent to the points that correspond to steps in the pattern of illumination (i.e. marked changes in intensity).

The retinal ganglion cells have been categorized into many groups based on different kinds of test. Hochstein and Shapley [5,6] proposed the classification of cells as X or Y in terms of the linearity or nonlinearity of their spatial summation. From their basic studies, the X-cells' spatial summation over their receptive field is approximately linear, while for Y-cells it is nonlinear. The X-cells respond to fine pattern and slow movement [7]. The Y-cells do seem to be involved in the detection of coarse pattern and fast movement or temporal change [5,6,7]. It should also be noticed that since Y-cells have a resolution limit of 40% of X-cells, it is extremely unlikely that they will signal the finest gratings in humans. However, the Y-cells are more numerous than the X-cells in peripheral fields.

II.1.2. Lateral Geniculate Nucleus

The lateral geniculate nucleus cells exhibit characteristics similar to those of the retinal ganglion cells [1]. The basic differences are : a considerable reduction in the spontaneous activity of X-cells at the lateral geniculate level, and an increase in the inhibition exerted by the surround on the center response of X-cells. Lateral geniculate cells give a smaller response to diffuse illumination than do retinal ganglion cells, sometimes giving no responses at all [8].

The X-cells of the retinal ganglion cells project to the lateral geniculate only, whereas the Y-cells project also to the superior colliculus (where sensory-motor integration takes place). Thus, combining the process of retinal ganglion cells and lateral geniculate nucleus, the X-cells are used for fine pattern analysis and slow motion detection, and the Y-cells for coarse pattern analysis and fast motion detection.

II.1.3. Distribution and shape of the receptive fields

It has been known for a long time that the spatial distribution of photoreceptors is nonuniform. The total response of those photoreceptors comprising a receptive field is sent to a ganglion cell. Thus, the increase in ganglion cell density is associated with an increase in the amount of visual information for unit visual angle transmitted to the higher level of visual processing. In the fovea region, the visual resolution (almost uniform) is very high which corresponds to more dense sampling (or higher density of ganglion cells). In the peripheral field, the visual resolution decreases with eccentricity¹. The resulting mapping from retina to striate cortex is topologically distorted, so the higher ganglion cell density correspond to larger areas in the striate cortex. This kind of mapping is called a retinotopic mapping.

The retinotopic mapping of the extrafoveal visual field to the surface of the striate cortex is characterized as a logarithmic conformal mapping [9]. The cortical magnification factor (the distance moved, in millimeters, across the cortical surface corresponding to one degree of movement of a point stimulus in the visual field) is radially symmetric and very close to an

¹Eccentricity is defined as the distance from the center of the fovea to the center of the receptive field.

inverse linear dependence on eccentricity for the entire peripheral visual field. For the central fovea region (less than 1 degree of eccentricity) the cortical magnification factor is almost a constant, and it is assumed that the inverse dependence on eccentricity tapers off in a gradual way. Therefore, the size of the receptive field increases with eccentricity whereas the density of ganglion cell decreases with eccentricity. The cortical magnification factor can be expressed by the following equation :

$$m = \frac{k}{r} \quad (2.1)$$

where k is a constant, and r represents eccentricity from the fovea. From this equation, a relation between the retina and striate cortex can be established as the complex logarithm mathematical formalism :

$$\frac{dw}{dz} = \frac{1}{z} \quad (2.2)$$

$$w = \ln(z) \quad (2.3)$$

where dw is a small change in cortical position, and dz is the corresponding small change in visual field position.

The magnification factor for the lateral-geniculate nucleus has the same form as that for the striate cortex. Cell density in both the lateral geniculate nucleus and the striate cortex is roughly constant with respect to eccentricity. Thus, the complex logarithmic structure of the striate cortex seems to be effected in two separate steps: 1) The form of the density of the retinal ganglion cells, leading to a logarithmic "radial" structure in the optic tract, and the lateral geniculate nucleus. 2) The projection of the lateral geniculate nucleus onto the cortex, where an angular reorganization of optic tract fibers

is effected which leads to the final form of the striate cortex retinotopic map as in the equation above.

Concerning the function of the retinal ganglion cell and the lateral geniculate nucleus receptive fields, a four mechanism model for edge detection has been established [10]. In [10], a model is presented for edge detection based upon probability summation among mechanisms with an excitatory center and an inhibitory surround. These mechanisms increase in size linearly with eccentricity. The four mechanisms are arbitrarily named N, S, T and U, the properties of each being defined by the difference of two Gaussian functions (DOG):

$$w_N(R, r) = A_N(R) \left[\exp \left[\frac{-r^2}{\sigma_N^2(R)} \right] - 0.57 \exp \left[\frac{-r^2}{(1.75\sigma_N(R))^2} \right] \right] \quad (2.4)$$

$$w_S(R, r) = A_S(R) \left[\exp \left[\frac{-r^2}{\sigma_S^2(R)} \right] - 0.57 \exp \left[\frac{-r^2}{(1.75\sigma_S(R))^2} \right] \right] \quad (2.5)$$

$$w_T(R, r) = A_T(R) \left[\exp \left[\frac{-r^2}{\sigma_T^2(R)} \right] - 0.30 \exp \left[\frac{-r^2}{(3\sigma_T(R))^2} \right] \right] \quad (2.6)$$

$$w_U(R, r) = A_U(R) \left[\exp \left[\frac{-r^2}{\sigma_U^2(R)} \right] - 0.2 \exp \left[\frac{-r^2}{(3\sigma_U(R))^2} \right] \right] \quad (2.7)$$

$$A_i(R) = \frac{A_i}{1 + a_i R} \quad , \quad i = N, S, T, U \quad (2.8)$$

$$\sigma_i(R) = \sigma_i(1 + kR) \quad , \quad i = N, S, T, U \quad (2.9)$$

In these equations the eccentricity R is measured in degrees, A_i (gain constant) and σ_i (space constant) are the values at the fovea, where a_i and k determine the rates of spatial variation, and r is the radius of the receptive field. Fig. II.3 shows the profile of the four mechanism at zero degrees (fovea) and at an eccentricity of 4 degrees.

N and S mechanisms are relatively sustained, T and U are relatively transient in their responses. The four mechanisms thus seem to show an appropriate reciprocity between spatial and temporal resolution with U being widest and most transient and N being narrowest and sustained.

Physiologically, the N and S mechanisms represent two different sizes of X-type cells, with the T and U mechanisms corresponding to Y-type cells.

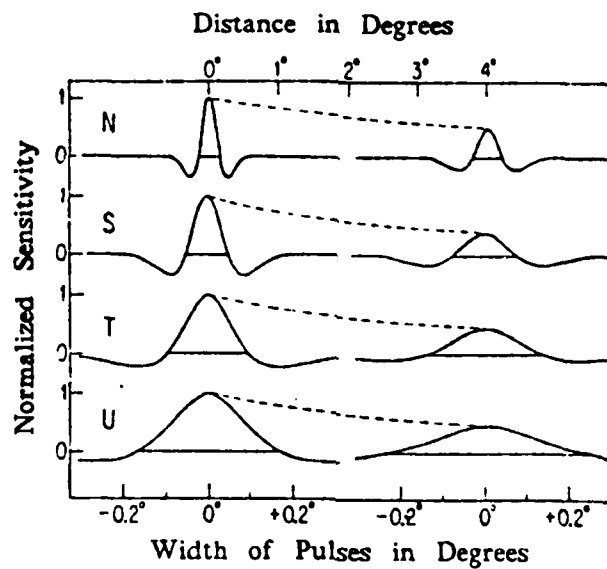


Fig. II.3. The profiles of the four mechanisms [10].

The N and S mechanisms could be classified as band-pass, while T and U mechanisms correspond to low-pass.

In general, N and S mechanisms (sustained) represent X-type cells with smaller receptive fields and function as fine edge and slow motion detection. T and U (transient) mechanisms represent Y-type cells with a larger receptive fields and function as coarse edge and fast movement detection, as mentioned in Sect. II.1.2. They all have a circularly symmetrical profile.

Summarizing, the main properties of the receptive fields of the peripheral ganglion cells are :

- (1) The size of the receptive field increases linearly with eccentricity whereas the density of the ganglion cells decreases linearly with eccentricity. This means that the resolution decreases with eccentricity or the sampling grid increases with eccentricity.
- (2) Overlapping between receptive fields.
- (3) The receptive field has a circularly symmetric and oscillating shape in which the center and the surround have opposite signs.
- (4) The magnification factor is radially symmetric, so the retinotopic mapping of the visual field to the surface of the striate cortex is characterized as a logarithmic conformal mapping.

II.2. Experiment on Peripheral Vision

This experiment was performed to establish the ability of human peripheral vision for moving object detection and shape determination.

The subject under testing was instructed to keep his gaze at a point marked on a white wall in front of him. The distance subject-wall was 1.5m. Trial runs were made with every subject until he became proficient at keeping his sight steady on the dot. The 2D geometric shapes used for testing had an average maximum side of 10cm. For example the hexagon has an 8.5cm side, the semicircle had a diameter of 10cm, etc.. The letters used for the recognition test were 10cm by 6cm. The number of subjects tested was five with an average age of 25 years, all male, and each test was performed twice with each subject, although the second time is less relevant than the first because the subject already knew what shapes to expect. Consequently, the tables below indicate the average for the five subjects using the first run only. The angle in tests 1 and 2, Tables II.1 and II.2, is measured as indicated in Fig. II.4(a). In Table II.1, notice that the negative angles for detection are average. Some subjects required small positive angles for detection. The first test was performed with the sample moving horizontally, i.e. on the XY plane, see Fig. II.4(a), while the second (Table II.3) was performed with the sample moving vertically along the XZ plane, Fig. II.4(b). The samples were attached to a radial spoke rotating on a shaft located above the subject's head and mounted on a frame. The velocity of the sample was, roughly, 2.6 degrees/second.

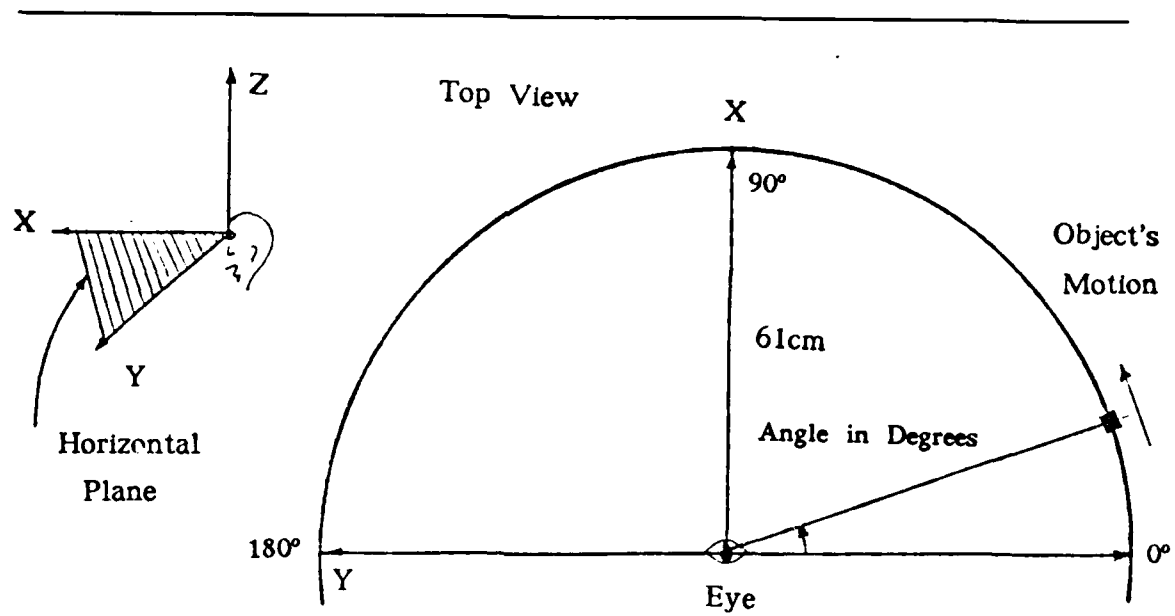


Fig. II.4(a) Measuring angle of detection for sample motion on the XY plane.

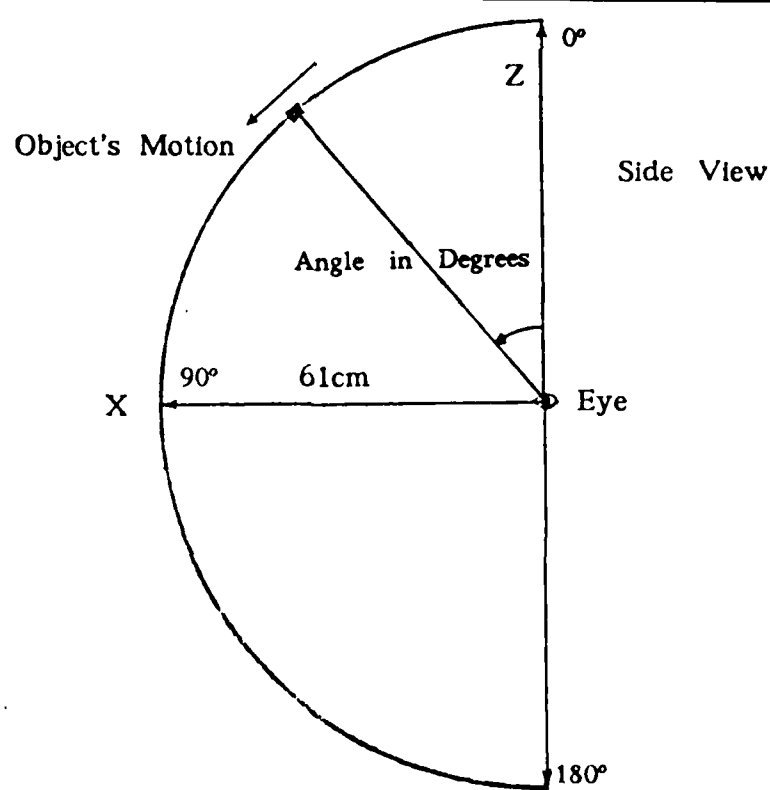


Fig. II.4(b) Measuring angle of detection for sample motion on the XZ plane.

Table II.1, The Objects Move on X-Y Plane. (Unit: Degree)

	detecting existence of the object	recognizing the shape
triangle	-3.2	18.2
circle	-2.6	19
rhombus	-2.5	20.25
square	-6	22.5
moon shape	-5.4	36.6
ellipes	-0.67	41.33
hexagon	-4	41.8
semi-circle	0	46
very complicated 3-D shape	-6.75	79.67

Note:

(1) The angle for detecting existence of the objects varies from -12 to 8 degrees.

(2) The resolution in angle is 1 degree, and data in the table are average values.

Table II.2, The Letters Move on X-Y Plane. (Unit: Degree)

letters	A	U	P	T	O	N	C
recognizing the letter	26.5	34.5	36.4	36.75	38	38	38
letters	D	E	B	F	Q	G	
recognizing the letter	40.75	44	44.5	45.2	50.8	51	

Notes:

(1) Maximum deviation for the letter recognition is 24.5°.

(2) It's difficult to distinguish between letters of similar shape.

Table II.3, The Objects Move on X-Z plane. (Unit: degree)

	detecting existence of the object	recognizing the shape	difference
triangle	41	47.25	6.25
square	41.25	50	8.75
moon-shape	36	52	16
circle	38.5	53.75	15.25
rhombus	38	54	16
semi-circle	35.67	58	22.33
hexagon	38.75	63.5	24.75
ellipes	37	66.33	29.33

II.2.1. Results

- (1) Peripheral vision is very sensitive to motion. It detects the moving object in far eccentricity very fast.
- (2) Peripheral vision extracts blurred or coarse edges and contours, because of the following reasons :
 - (a) The sampling distance (spacing between centers of adjacent receptive fields) increases with eccentricity.
 - (b) As the radius of the receptive field is getting larger with eccentricity, most of the low frequency components of the image are passed by the processing of the peripheral visual system.
 - (c) Peripheral vision has much wider detecting range in the horizontal direction than in the vertical direction.

II.3. Simulation of Edge Detection in the Human Visual System

Preprocessing performed by the peripheral visual system (PVS) includes: edge detection and motion detection which could be represented by the block diagram of Fig. II.5.

II.3.1. Scheme of the geometric distribution of the receptive fields of the retinal ganglion cells and the lateral geniculate nucleus

According to the main properties of the receptive fields of the peripheral visual system, the scheme of the geometric distribution of the receptive fields of the retinal ganglion cells and the lateral geniculate nucleus, which Braccini [1,11,12] uses, is appropriate for simulation but with different organization. It is shown in Fig. II.6. The geometric centers of the circles represent the

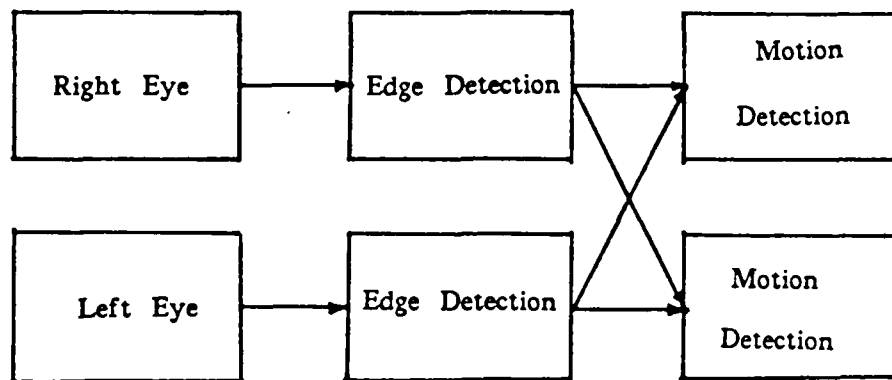


Fig. II.5. Block diagram of preprocessing in human visual system.

sampling points and form a triangular lattice. The diameter of the circles is proportional to the receptive field size. Both the sampling distance and the receptive field size increase linearly with eccentricity.

II.3.2. Weighting functions of the receptive field

It was mentioned in Sect. II.1.3 that four mechanisms (N, S, T and U) have been postulated. For edge detection, the N mechanism has been chosen, as the most significant, for simulation:

$$w_N(R, r) = \exp\left[\frac{-r^2}{\sigma_N^2(R)}\right] - 0.57 \exp\left[\frac{-r^2}{(1.75\sigma_N(R))^2}\right] \quad (2.10)$$

The $A_N(R)$ in (2.4) only changes the amplitude and in (2.10) it has been normalized to 1, because for edge detection only contrast is important. The

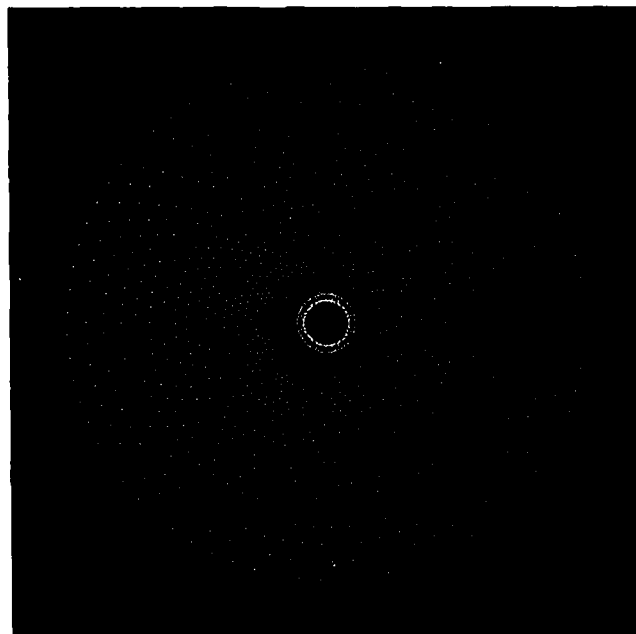


fig. II.6. The scheme of the geometric centers of the receptive fields.

above equation can be approximated [13,14] as follows,

$$w = \left(1 - \frac{r^2}{p^2}\right) \exp\left(\frac{-r^2}{2p^2}\right) \quad (2.11)$$

$$p = 0.92\sigma(1 + kR)$$

where R is the eccentricity, r is the radius of the receptive field and k is a constant factor to weight the influence of R , [10]. The exact DOG function (2.10) and approximated function (2.11) produce very close results, see Fig. II.7.

II.3.3 Convolution

The processing performed by the retinal ganglion cells and the lateral geniculate nucleus can be approximated by the convolution of the input image and the weighting function $w(R,r)$:

$$g(R,\phi) = \int_0^\infty \int_{-\pi}^\pi w(R,r) f(R \cos \phi - r \cos \theta, R \sin \phi - r \sin \theta) r dr d\theta \quad (2.12)$$

where $f(R,\phi)$ is the input image which is formed in the retina, $w(R,r)$ is the weighting function of (2.11) and $g(R,\phi)$ is the output image, R and r have already been defined and ϕ, θ are indicated in Fig. II.9(a). The convolution output $g(R,\phi)$ for constant input $f(R,\phi) = F$ is $-2\pi F p^2$, if the above weighting function $w(R,r)$ is used. Because the convolution is not zero for constant input, it is difficult to extract the exact edge for further image processing. It is desired to have a zerocrossing to unambiguously indicate the presence of an edge in the image. To avoid this, the weighting function (2.11) is modified as follows:

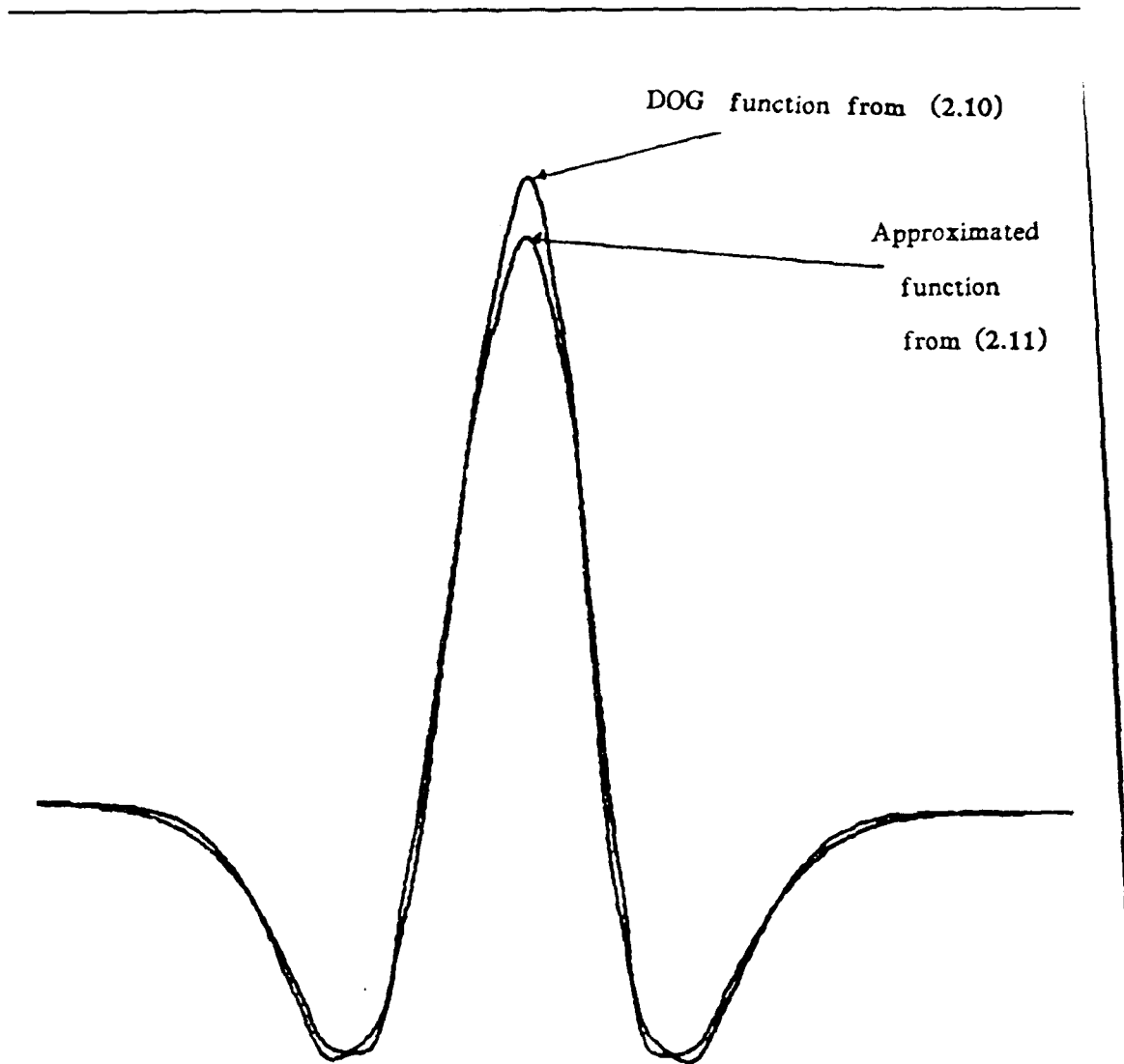


Fig. II.7. The exact and the approximated DOG functions.

$$w = \left(2 - \frac{r^2}{p^2} \right) \exp \left(\frac{-r^2}{2p^2} \right) \quad (2.13)$$

where $p = (1 + kR)\sigma$; (2.13) has the same form as the Laplacian of Gaussian function presented by Marr and Hildreth [8,16]. The above weighting function is space variant with eccentricity and the convolution output $g(R, \phi)$ is

zero for constant input $f(R,\phi)=F$.

In the Gaussian function of Marr and Hildreth,

$$G(r)=\sigma^2 \exp\left(\frac{-r^2}{2\sigma^2}\right) \quad (2.14)$$

σ is the standard deviation for a given spatial location. A way to generalize this function to give a spatially variant weighting function is to replace σ by $(1+kR)\sigma$ and also change the sign, so

$$w=\nabla^2 G_{space-variant}=\left(2-\frac{r^2}{p^2}\right) \exp\left(\frac{-r^2}{2p^2}\right) \quad (2.15)$$

where $p=(1+kR)\sigma$. And according to the property that the convolution of the Laplacian-of-the-Gaussian function with the input is equal to the Laplacian of the convolution of the two functions,

$$g(R,\phi)=\nabla^2 G_{space-variant} * f(R,\phi)=\nabla^2 \left[G_{space-variant} * f(R,\phi) \right] \quad (2.16)$$

So, whenever an intensity change occurs, there will be zero-crossings in those intensity changing points. Thus, detecting the change of intensity (edge) can be reduced to finding the zero crossings, see Fig. II.8.

II.3.4. Simulation procedure and results

According to the scheme of the geometric distribution of the receptive fields which is shown in Fig. II.6, the discrete convolution of input image and the weighting function of the receptive field is applied to each sampling point for edge detection simulation.

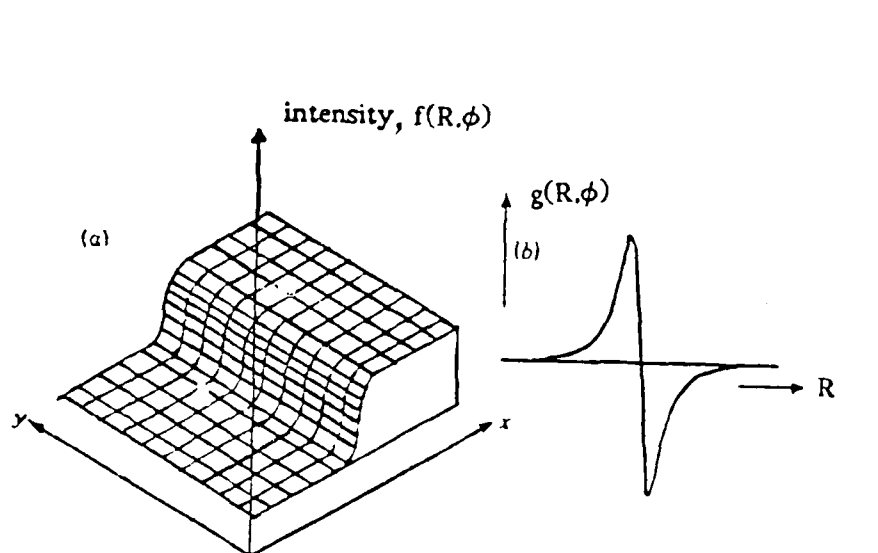


Fig. II.8. (a) Change of intensity in two dimensions (b) Zero-crossing for a radial cross section [8].

II.3.4.1. Procedure

- (1) Calculate the sampling grid (according to Fig. II.6)

Before calculating the sampling grid, the smallest eccentricity (the radius of the fovea), the smallest radius of the central region of the overlapping receptive field and the overlapping factor must be given. Then, the N, S, T and U mechanisms could be simulated by giving different values to the smallest radius of the central region of the overlapping receptive fields.

(a) overlapping factor :

AD-A160 521 BIOLOGICAL VISUAL SYSTEMS STRUCTURES FOR MACHINE VISION 2/4

24

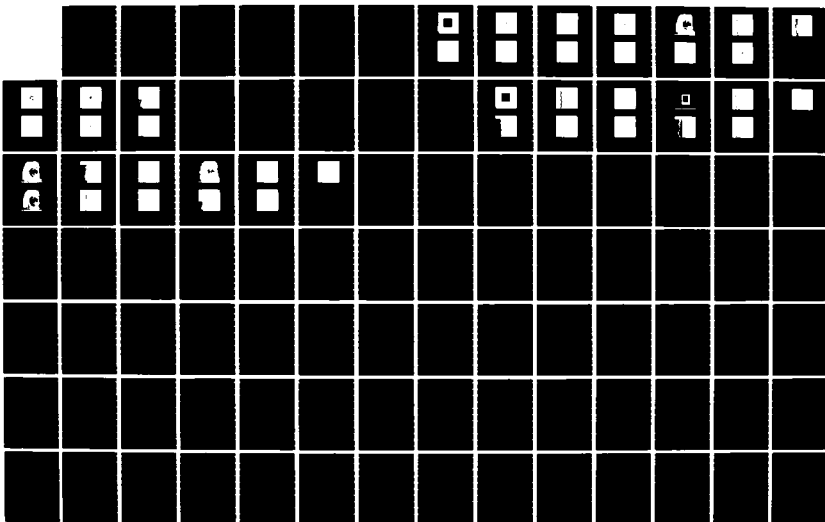
APPLIED TO ROBOTI.. (U) VIRGINIA UNIV CHARLOTTESVILLE

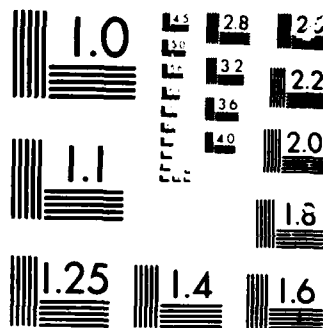
DEPT OF ELECTRICAL ENGINEERING.. R M INIGO ET AL

UNCLASSIFIED FEB 86 UVA/525647/EE86/101 AFOSR-TR-86-0282 F/G 6/4

FEB 86 UVA/525647/EE06/101 AFOSR-TR-86-0282 F/G 6/4

REL





$$of = \frac{r_{o_i} - r_{no_i}}{r_{o_i}} \quad (2.17)$$

$$r_{no_i} = (1 - of) r_{o_i} \quad (2.18)$$

where

r_{o_i} : the radius² of the elements in ring " i " for an overlapping receptive field.

r_{no_i} : the radius² of the elements in ring " i " for a non-overlapping receptive field.

(b) Number of elements per ring:

$$N = \frac{\pi}{\arcsin \frac{r_{no_i}}{R_i}} \quad (2.19)$$

$$\psi = \frac{\pi}{N} \quad (2.20)$$

where R is eccentricity from fovea.

(c) The relationship for eccentricity between rings is, see Fig. II.9(a),

$$\frac{R_{i+1}}{R_i} = A + \sqrt{A^2 - 1} \quad (2.21)$$

$$A = \tan^2 \psi + \frac{1}{\cos \psi} \quad (2.22)$$

(d) The relationship between the eccentricity and the radius of the central region for a non-overlapping receptive field:

²These radii refer to the dimension for the positive part of the Laplacian-Gaussian function. The radii for the negative part are actually larger.

$$r_{no_i} = R_i \sin \psi \quad (2.23)$$

By starting with given values for R_1 , r_{o_1} and the overlapping factor, the values of rings and the number of elements per ring can be calculated, and also their dimensions, using the above relations.

(2) Discrete convolution

Since the sampling grid is discrete, the value of $f(R \cos \phi - r \cos \theta, R \sin \phi - r \sin \theta)$ is only a function of θ in $0 \leq r \leq 1$, $1 \leq r \leq 2$, $2 \leq r \leq 3$, etc. Thus,

$$\begin{aligned} g_{discrete}(R, \phi) = & 2\pi F_{00} \int_0^{0.5} w(R, r) r dr \\ & + \nabla \theta_1 (F_{10} + F_{11} + F_{12} + F_{13} + \dots + F_{17}) \int_{0.5}^{1.5} w(R, r) r dr \\ & + \nabla \theta_2 (F_{20} + F_{21} + F_{22} + F_{23} + F_{24} + F_{25} + F_{26} + F_{27} \dots) \int_{1.5}^{2.5} w(R, r) r dr + \dots \end{aligned} \quad (2.24)$$

where the F's have been indicated in Fig. II.9(b).

For

$$\nabla \theta_1 = \frac{2\pi}{8}$$

$$\nabla \theta_2 = \frac{2\pi}{12}$$

and normalizing to 2π ,

$$\begin{aligned} g_{discrete}(R, \phi) = & F_{00} \int_0^{0.5} w(R, r) r dr \\ & + \frac{1}{8} (F_{10} + F_{11} + F_{12} + F_{13} + \dots + F_{17}) \int_{0.5}^{1.5} w(R, r) r dr \\ & + \frac{1}{12} (F_{20} + F_{21} + F_{22} + F_{23} + F_{24} + F_{25} + F_{26} + F_{27} \dots) \int_{1.5}^{2.5} w(R, r) r dr + \dots \end{aligned} \quad (2.25)$$

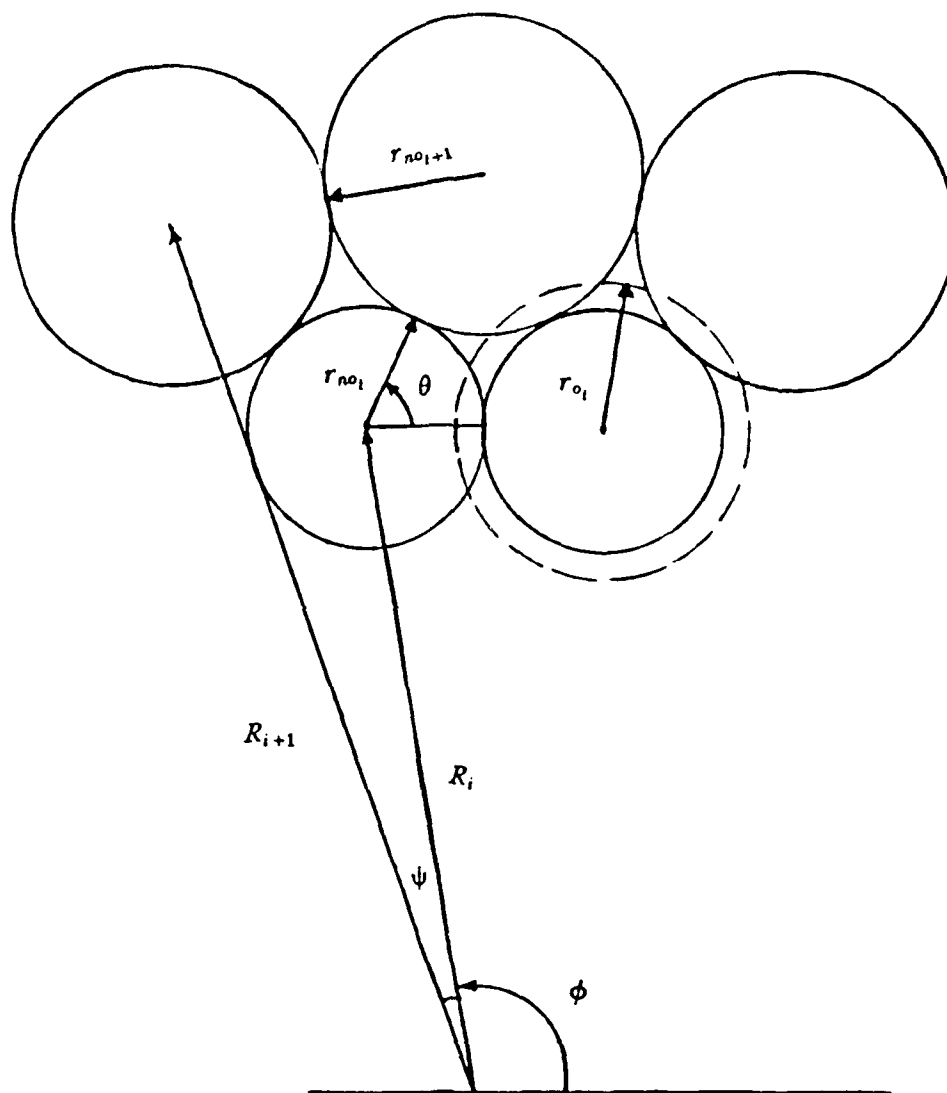


Fig. II.9(a). Geometry of sampling grid (concentric rings).

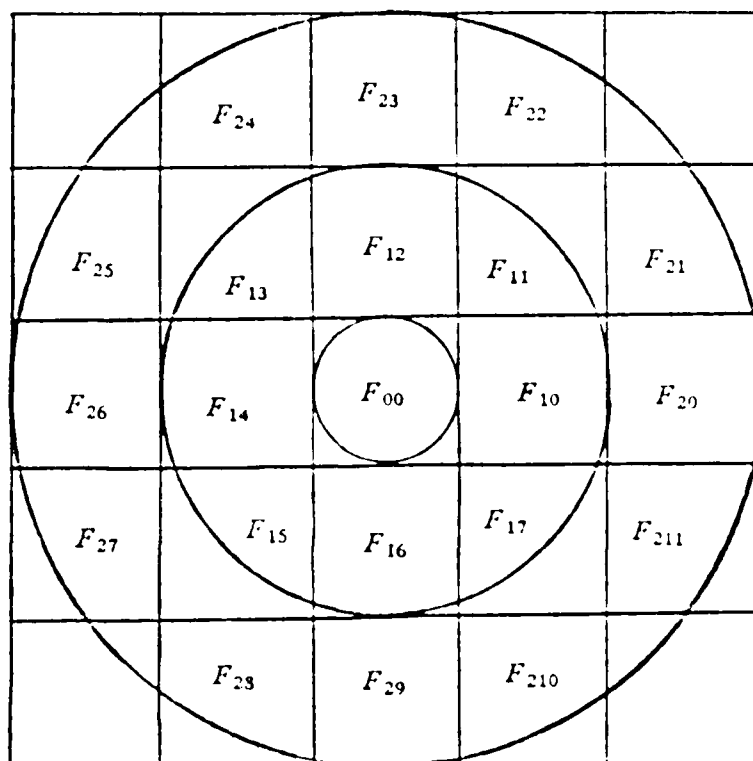


Fig. II.9(b). Geometry for simulation using rectangular pixels.

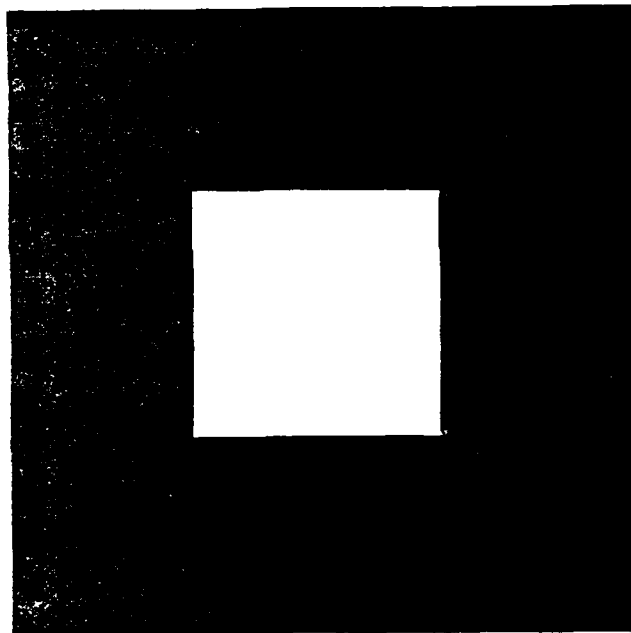
$$\int w(R, r) r dr = r^2 \exp \left(\frac{-r^2}{2p^2} \right) \quad (2.26)$$

where $p = \frac{r_{o1}}{\sqrt{2}}$, and the dependence on R is implicit because $r_{o1} = f(R)$. If the upper limit of the integration is selected as $r = 4.5r_o$ instead of $r = \infty$, the error

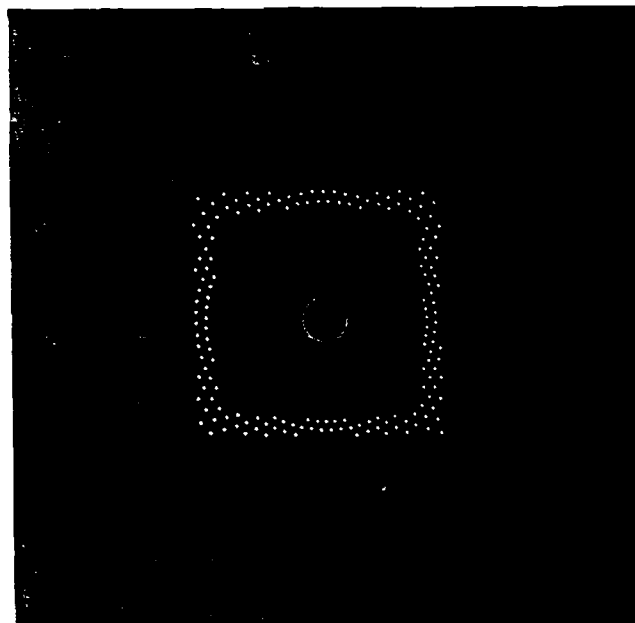
is very small.

II.3.4.2. Results

Edge detection simulation by means of the above procedure was performed using a white square on black background, Fig. II.10(a), and a noisy photograph of a woman's face in gray level, Fig. II.12(a). Figs. II.10(c),(d) and II.11(c),(d) show the edges of the square. Figs. II.12(d),(e) and II.13(c),(d),(e),(f) show the edges of the woman's face. The number of sampling points indicated in Figs. II.10 and II.12 is 71 by 32 by choosing $r_{o_1}=1$ pixel, $R_1=18$ pixels and of=0.2 and in Figs. II.11 and II.13. is 113 by 51 by choosing $r_{o_1}=1$ pixel, $R_1=18$ pixels and of=0.5. The convolution for the original images produces the positive values, negative values and zero values. The real edges are the zero-crossing points. The edges are not quite straight lines in Fig. II.10, especially at corners, because of the discrete nature and small number of the sampling points. Since the distance between sampling points increases linearly with eccentricity, this is a non-uniform sampling case, and the greater the distance between sampling points, the larger the error. When the number of sampling points is increased, as in Fig. II.11, the results are much better. This characteristic coincides with the experiment on peripheral vision. In the PVS experiment, the object which is perceived at greater eccentricity is more distorted. The images of the lady's face in Fig. II.12 and II.13 are very noisy, but the resulting images are still recognizable. Although this simulation procedure only provides a rough or approximated edge of an image and the computation is complicated, it compresses the original image data (and thus saves memory space) for further processing, such as motion detection, consequently saving computing time.

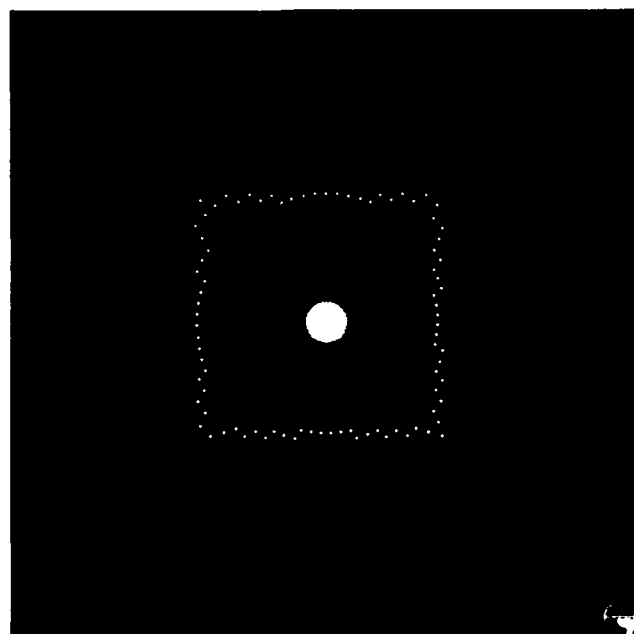


(a) Original image: white square on black background

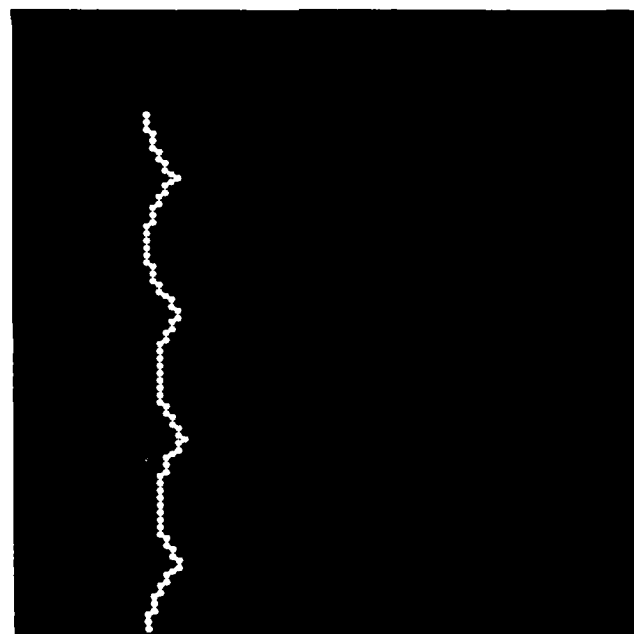


(b) Image after edge detection in image plane: bright points are positive values; dark points are negative values.

Fig.II.10 Edge Detection Simulation Using HVS Detection Model For 71 by 32 Sampling Points.

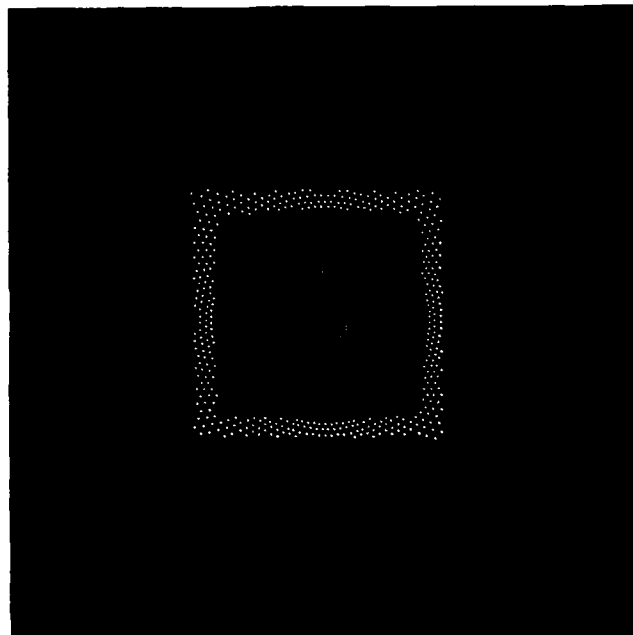


(c) Zero-crossing points (real edges) in image plane

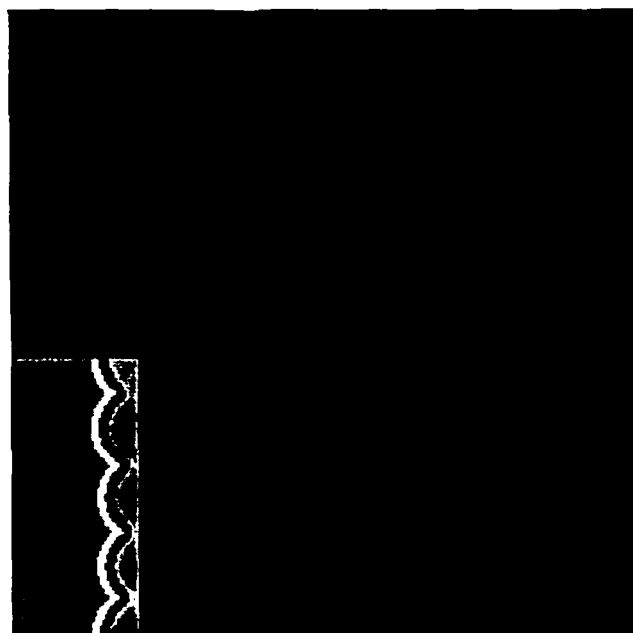


(d) Real edges in computation plane

Fig.II.10 Edge Detection Simulation Using HVS Detection Model For 71 by 32 Sampling Points

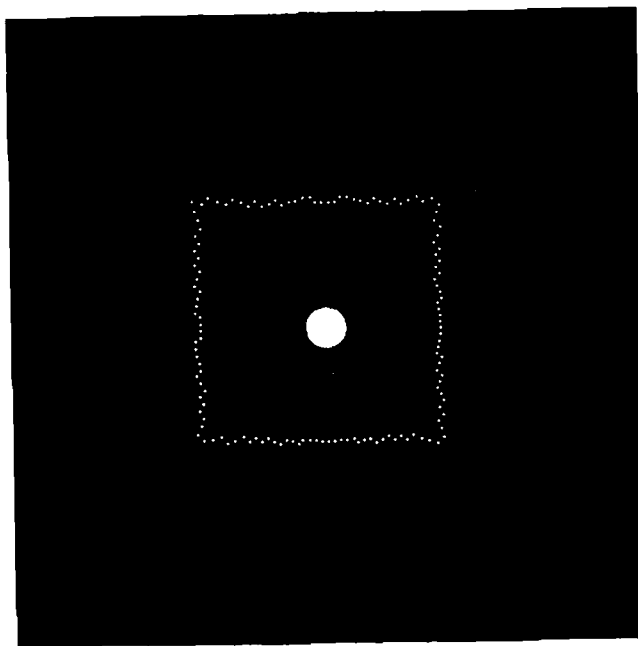


(a) Image after edge detection in image plane: bright points are positive values; dark points are negative values

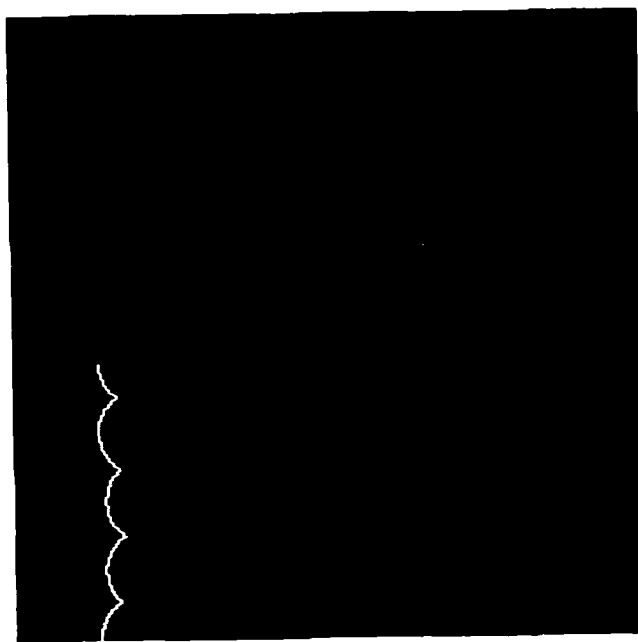


(b) Image after edge detection in computational plane

Fig.II.11 Edge Detection Simulation Using HVS Detection Model for 113 by 51 Sampling Points.



(c) Zero crossing points (real edges) in image plane

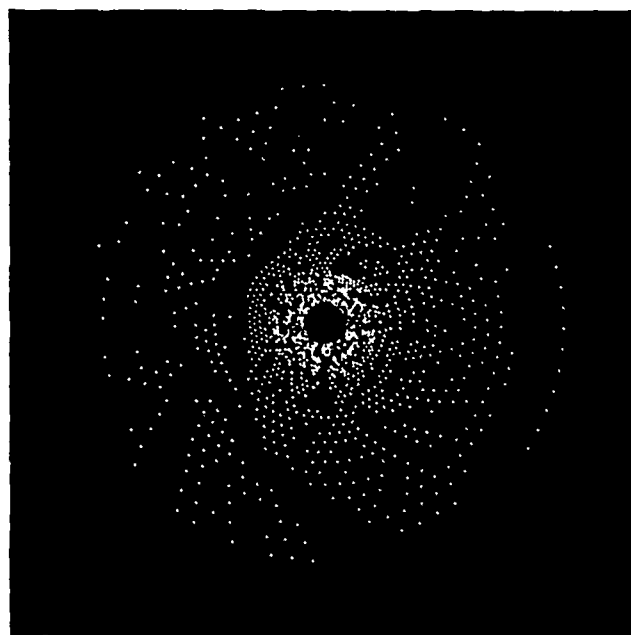


(d) Real edges in computation plane

Fig.II.11 Edge Detection Simulation Using HVS Detection Model for 113 by 51 Sampling Points.

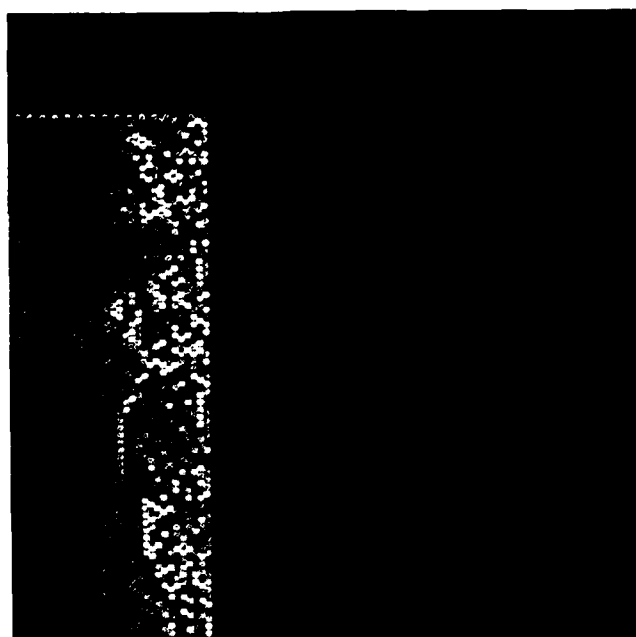


(a) Original image: noisy photograph

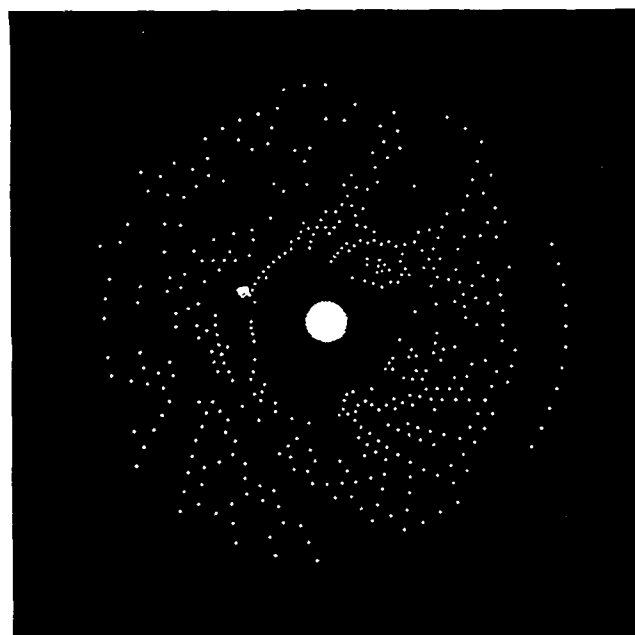


(b) Image after edge detection, image plane

Fig.II.12 Edge Detection Simulation Using HVS Detection Model For 71 by 32 Sampling Points.

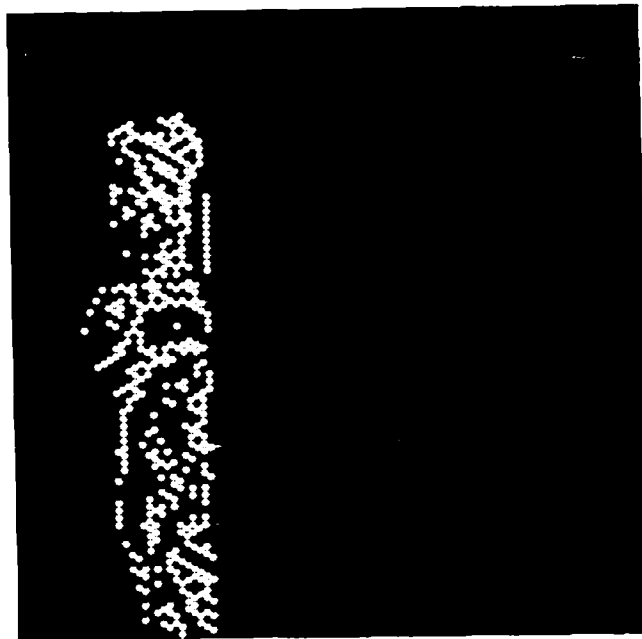


(c) Image after edge detection, computation plane



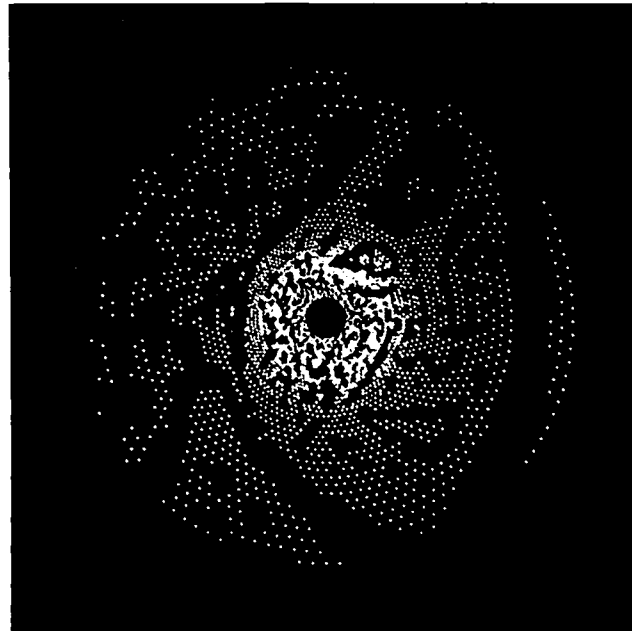
(d) Zero crossing edges, image plane

Fig.II.12 Edge Detection Simulation Using HVS Detection Model For 71 by 32 Sampling Points.

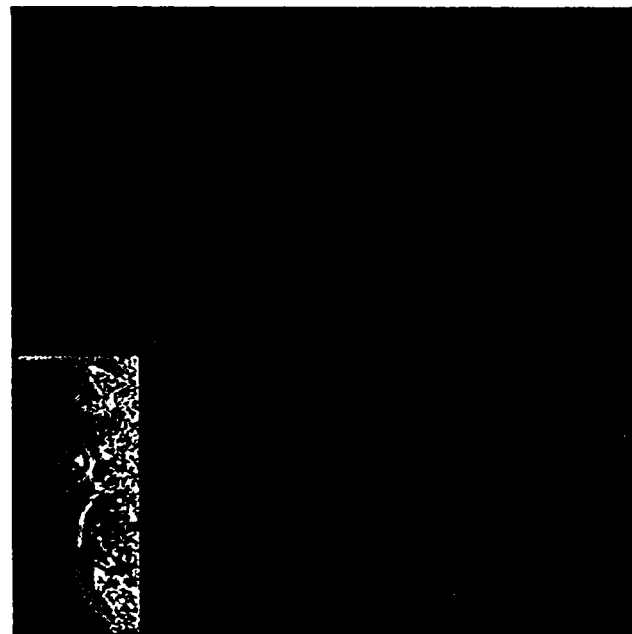


(e) Zero crossing edges, computation plane

Fig.II.12 Edge Detection Simulation Using HVS Detection Model For 71 by 32 Sampling Points.

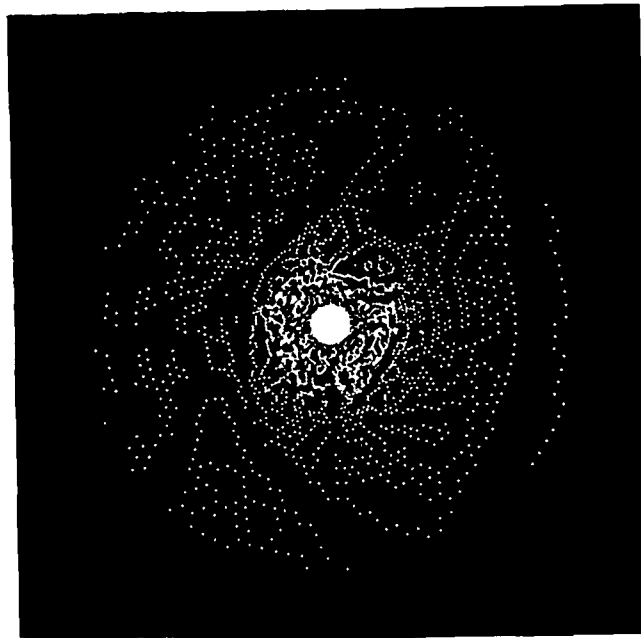


(a) Image after edge detection, image plane



(b) Image after edge detection, computation plane

Fig.II.13 Edge Detection Simulation Using HVS Detection Model for 113 by 51 Sampling Points

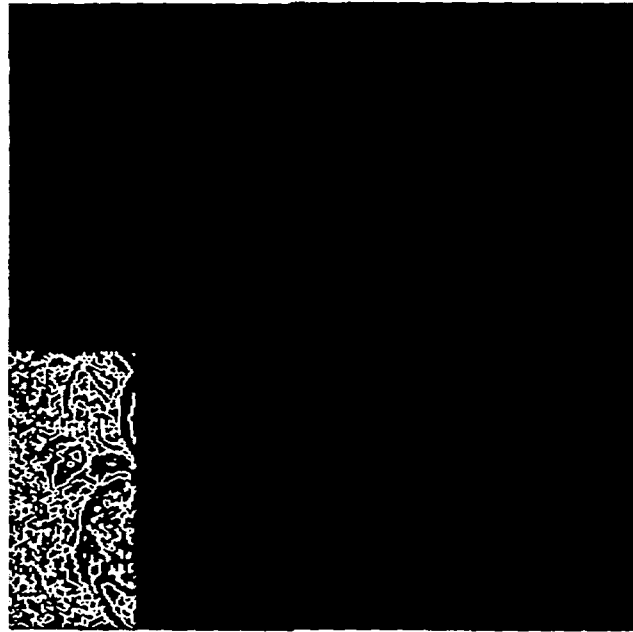


(c) Zero crossing edges, image plane, no thresholding

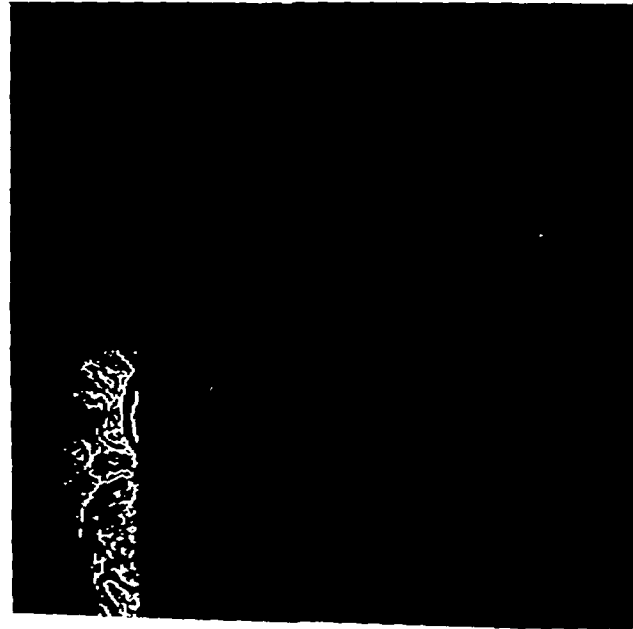


(d) Zero crossing edges, image plane, threshold = 20

Fig.II.13 Edge Detection Simulation Using HVS Detection Model for 113 by 51 Sampling Points



(e) Zero crossing edges, computation plane, no thresholding



(f) Zero crossing edges, computation plane, thresholding = 20

Fig.II.13 Edge Detection Simulation Using HVS Detection Model for 113 by 51 Sampling Points

II.4. Edge Detection Simulation for New Sensor

As indicated in Chapt. 1, there are three basic geometrics for the new sensor:

- (1) Arcs of ring
- (2) Circular elements
- (3) Hexagonal elements

The image is acquired by the new sensor's image plane and the edge detection and motion detection is accomplished in computation plane. Fig. II.14 explains the relationship.

For the arcs of ring sensor, the mapping in computation plane is a rectangular array, so the edge detection in this plane can use a variety of methods, such as the local neighborhood methods (Roberts operator, Prewitt operator and Sobel operator etc.), regional methods (Laplacian Gaussian), global methods (signal analysis) etc. [17,18].

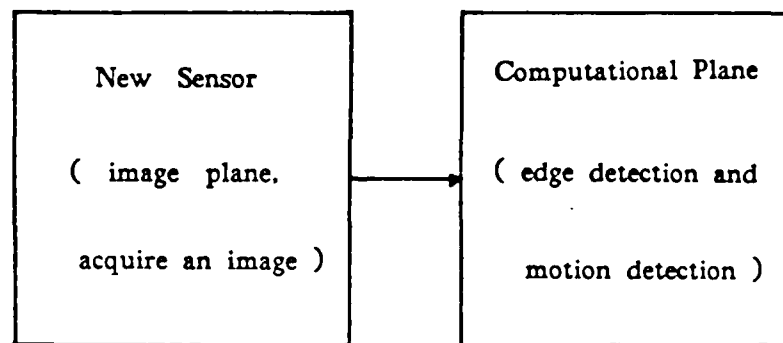


Fig. II.14. Block diagram of image processing procedure for the new sensors.

For the circular and hexagonal sensors, the mapping to computation plane is staggered, so another method is needed to extract the edge. The following local neighborhood methods to detect edge were used to determine the gradient at pixel a_1 , refer to Fig. II.15.

(1) Absolute value method:

$$g = |a_1 - a_2| + |a_1 - a_3| + |a_1 - a_4| + |a_1 - a_5| + |a_1 - a_6| + |a_1 - a_7| \quad (2.27)$$

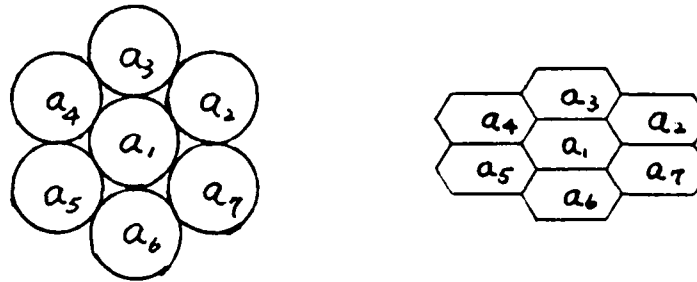


Fig. II.15. Circular elements and hexagonal elements neighborhoods in computation plane.

(2) Laplacian method: (See Fig. II.16)

$$g = |a_1 - a_2| + |a_1 - a_3| + |a_1 - a_4| + |a_1 - a_5| + |a_1 - a_6| + |a_1 - a_7| \quad (2.28)$$

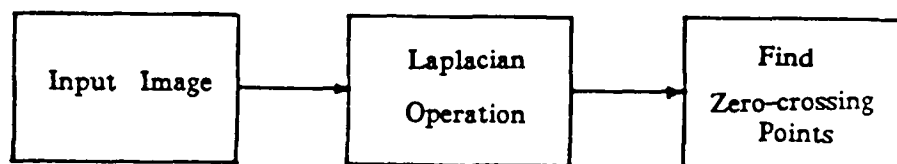


Fig. II.16. Block diagram of the Laplacian operation.

(3) Using the Laplacian of Gaussian function.

The general edge detection scheme is indicated in the block diagram of Fig. II.17.

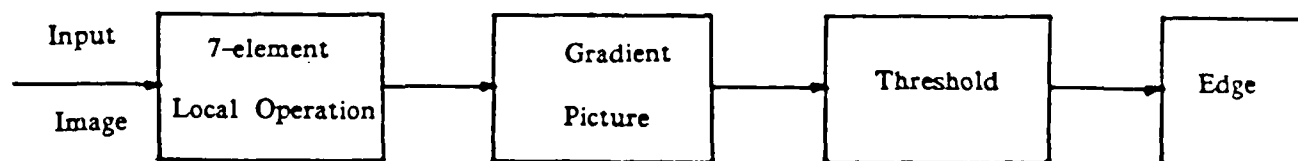


Fig. II.17. Block diagram of the general edge detection scheme.

II.4.1. Simulation procedure and results

II.4.1.1. Input the image from circular sensor

Input the image from the image plane of the circular sensor. Edge detection is implemented in computational plane. The examples of this simulation are shown in Figs. II.18,19,20 and 21. The original image in Fig. II.18 is binary and the original images in Figs. II.20 and 21 are very noisy pictures. From the examples, the absolute value method produces thick edges, so a thinning operation should be used. For non-binary pictures, the absolute value and Laplacian methods need thresholding, but not necessarily the Laplacian-Gaussian method. The absolute value method and Laplacian method are very fast and simple for edge detection. The Laplacian-Gaussian function is slow and complicated for edge detection, but it produces good results for noisy images.

II.5 Savings in Computations for the New Sensor Compared to Rectangular Sensor

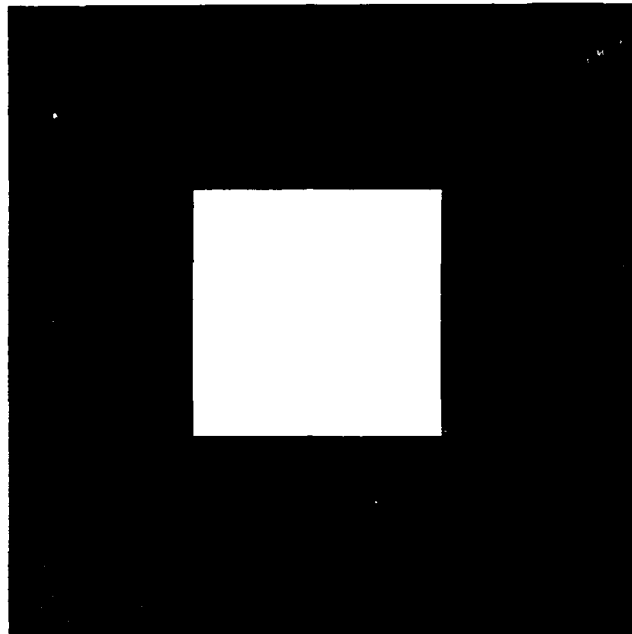
Because of the small number of elements in the new sensor compared to a conventional rectangular array, the number of computations for image processing is reduced dramatically. For example, edge detection using the absolute value method or the Laplacian method needs 11 additions per pixel, but the total number of operations is less than using the Sobel's edge operator for the rectangular sensor. The computational time for edge operation of the new sensor using 71 by 34 elements will be only one hundredth compared to that of the rectangular sensor using 512 by 512 pixels, and, as can be seen from the edge image of Fig. II.18(a), for example, the main features of

the image should still be recognizable.

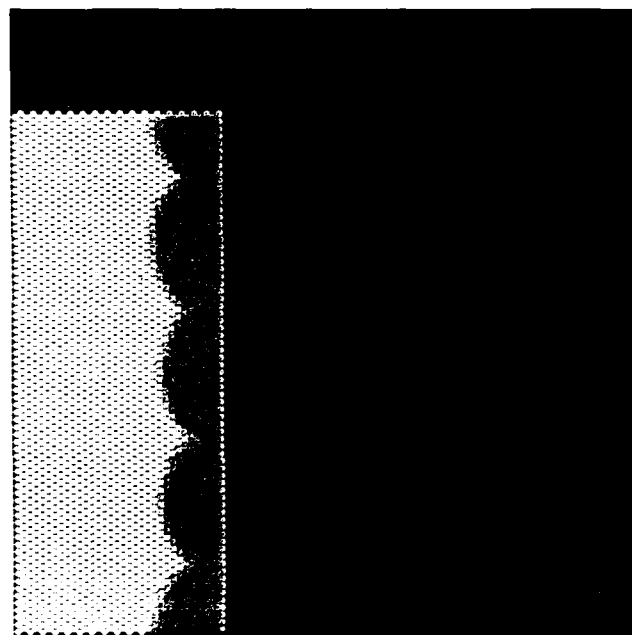
II.6. Comments and Conclusion

In the previous sections, we have presented ,and illustrated by means of examples, a mathematical model of processing performed by the peripheral visual system and the new sensor, based on the properties discussed in Sect. II.1 and Sect. II.4.

Although the simulation of edge detection for the human PVS proves the existence of preprocessing, it might be difficult to implement it in hardware and more theoretical research is necessary before a working model for hardware implementation is developed. Comparing these two simulations, the results are very similar. The image plane-computation plane sensor based in the HVS should be simpler to implement in hardware but, as mentioned in Chapt. II, more research is necessary on the physical lay-out and also on the effects of resolution reduction on pattern recognition techniques and on how to apply established a new pattern recognition techniques to images in the computation plane.

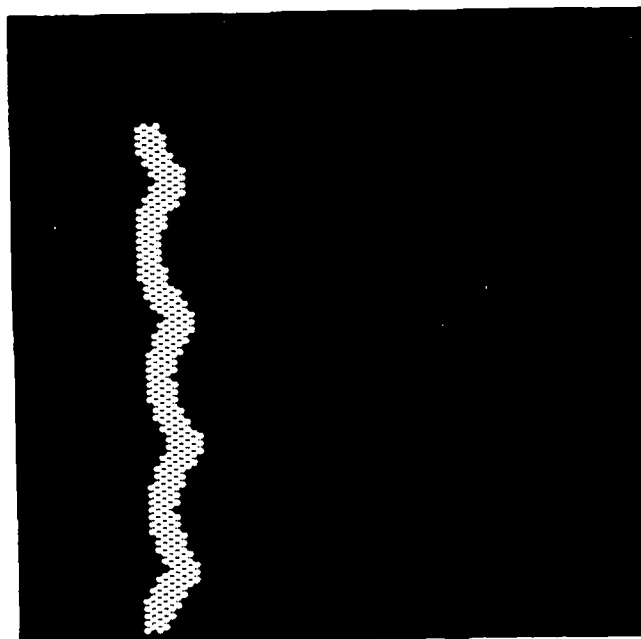


(a) Original image: white square on black background

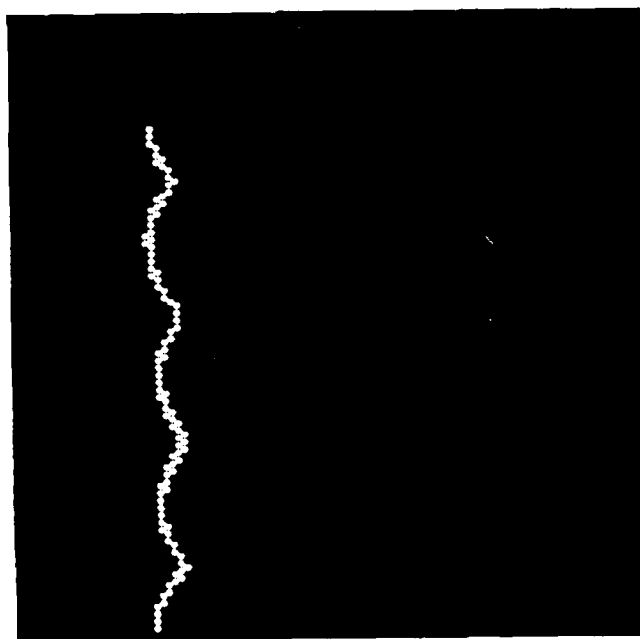


(b) Mapping of image to computation plane

Fig.II.18 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings 74 Elements per Ring).

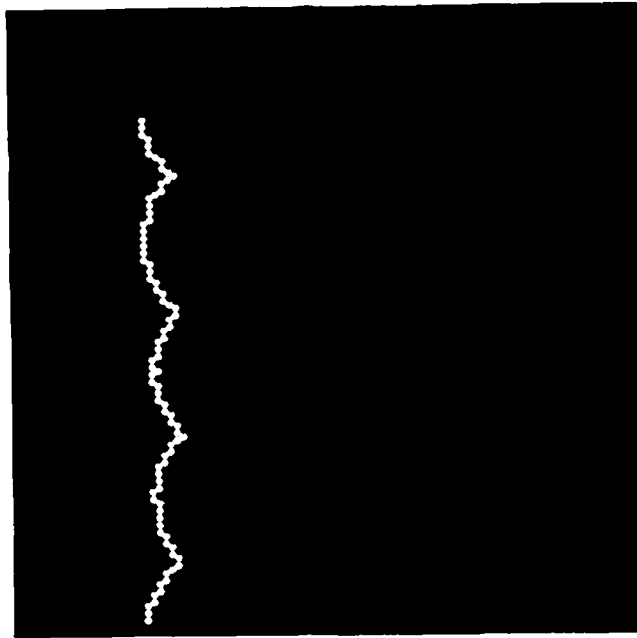


(c) Edge detection in computation plane by absolute value method and no thresholding

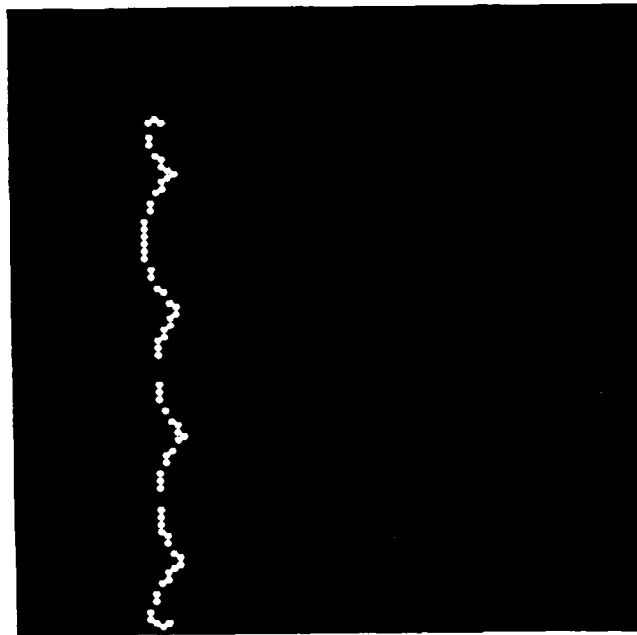


(d) Edge detection in CP by absolute value method; threshold = 120

Fig.II.18 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings 74 Elements per Ring).

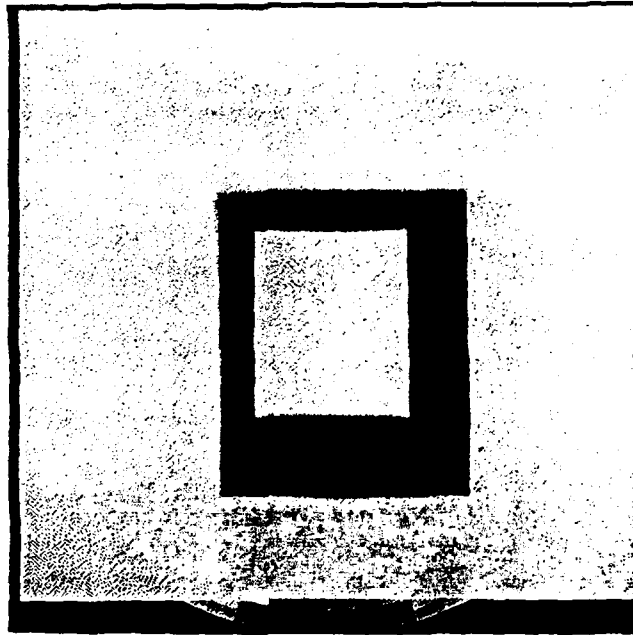


(e) Edge detection in CP by Laplacian method, no thresholding

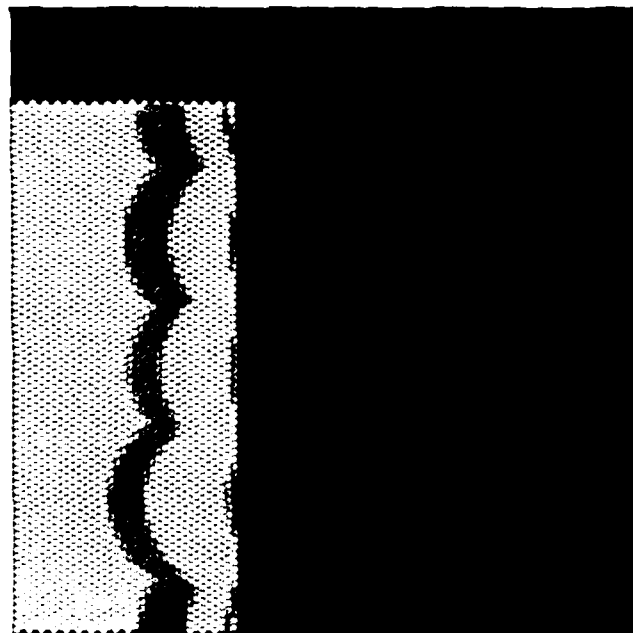


(f) Edge detection in CP by Laplacian-Gaussian method, no thresholding

Fig.II.18 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings 74 Elements per Ring).

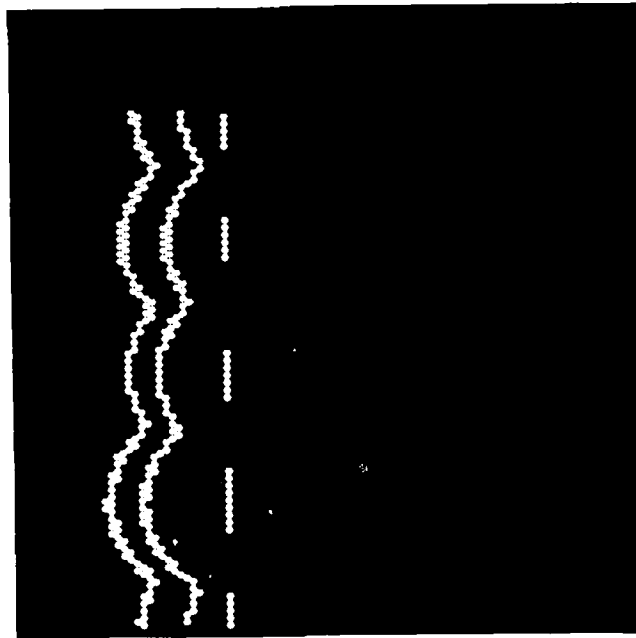


(a) Original grey level image: dark frame on light background

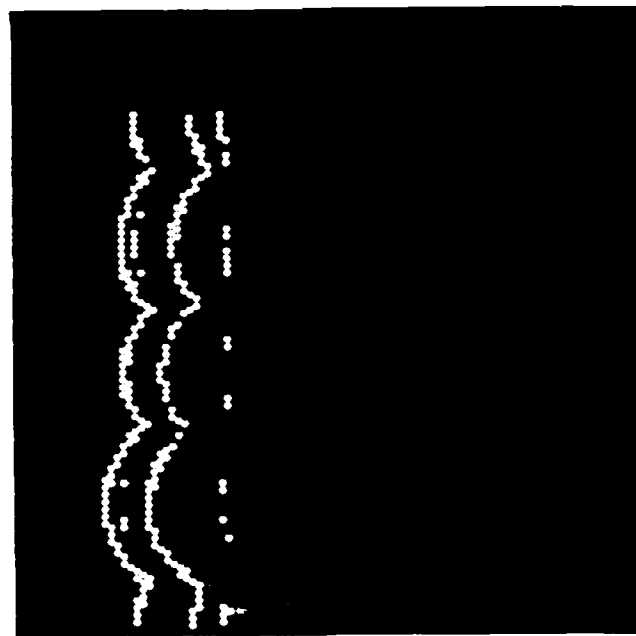


(b) Mapping of image to CP

Fig.II.19 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).

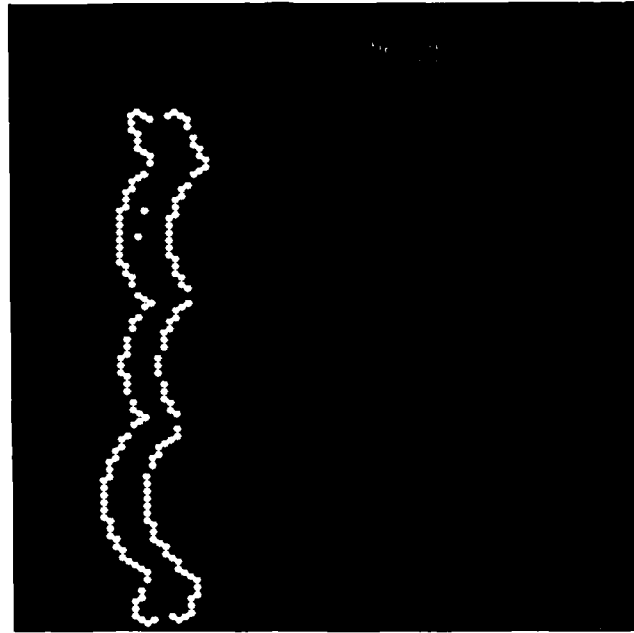


(c) Edge detection in CP by absolute value method; threshold = 190



(d) Edge detection in CP by Laplacian method; threshold = 30

Fig.II.19 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).



(e) Edge detection in CP by Laplacian-Gaussian method, no thresholding

Fig.II.19 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).

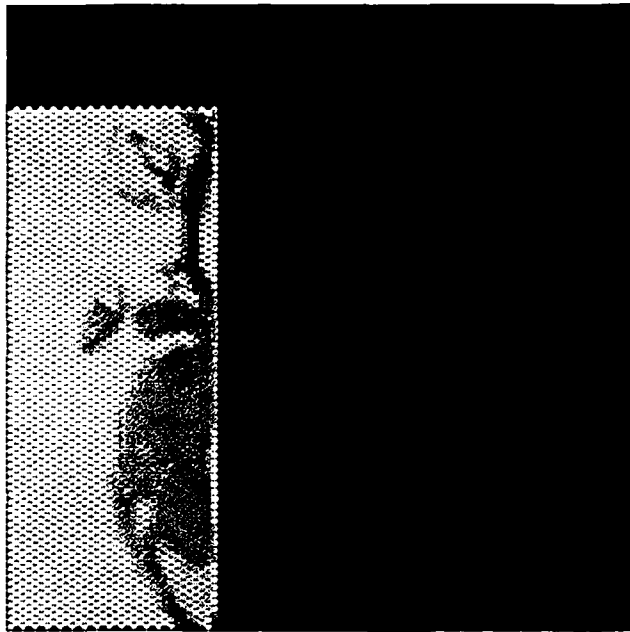


(a) Original image

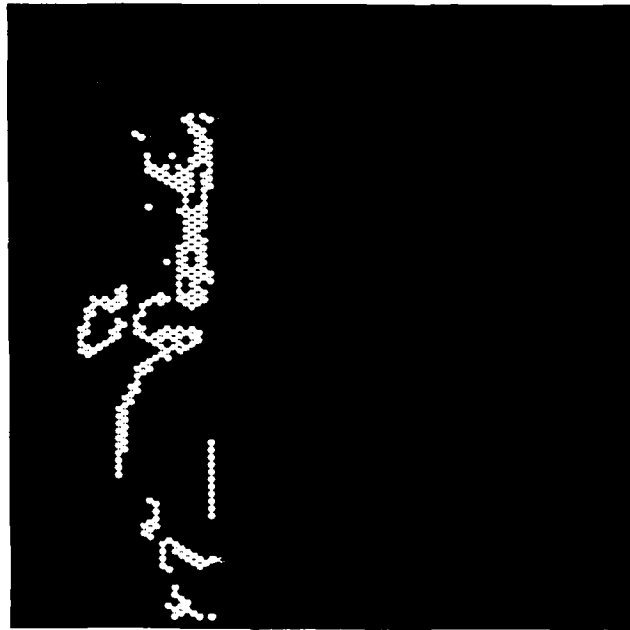


(b) Simulated circular-element sensor image

Fig.II.20 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).

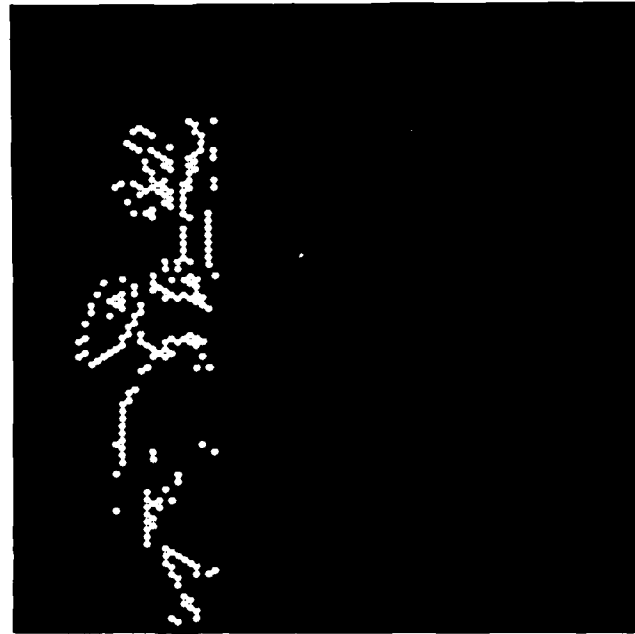


(c) Mapping of image to computation plane

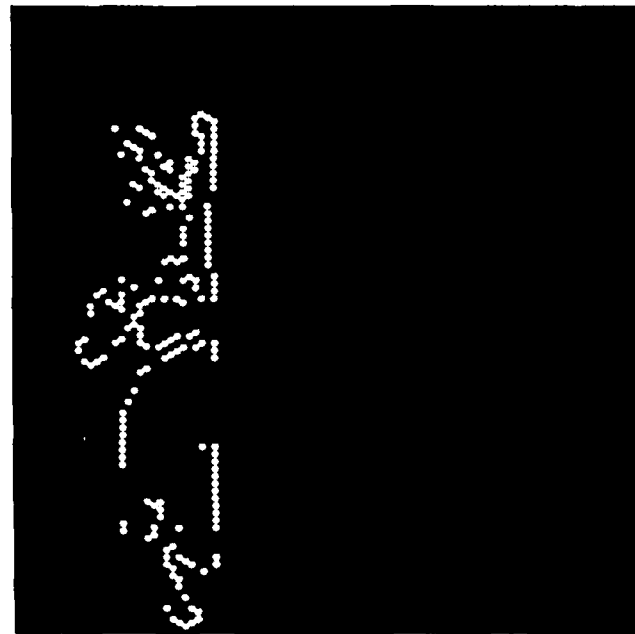


(d) Edge image in CP, absolute value method, threshold = 75

Fig.II.20 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).



(e) Edge image in CP, Laplacian method, threshold = 20

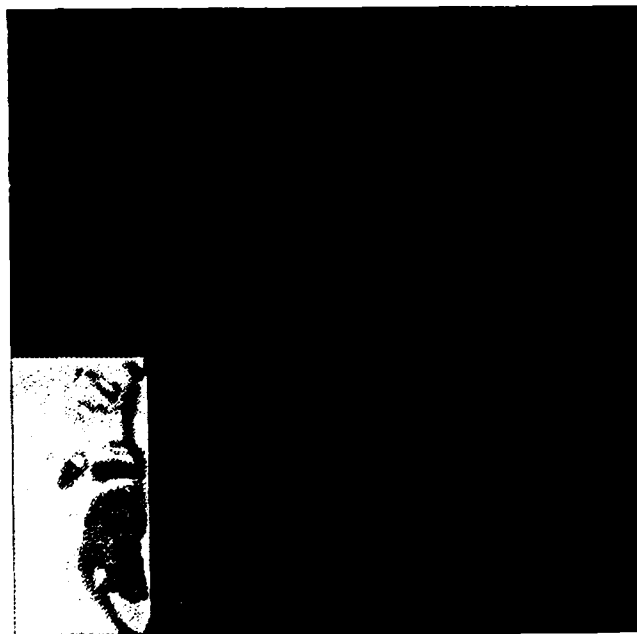


(f) Edge image in CP, Laplacian-Gaussian method, no thresholding

Fig.II.20 Edge Detection Using Simulation of Circular-Elements Sensor (34 Rings, 74 Elements per Ring).

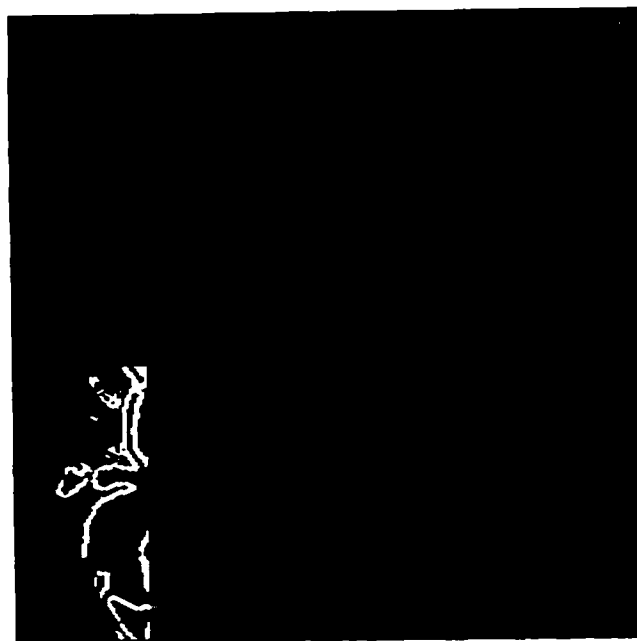


(a) Simulated circular-element sensor image

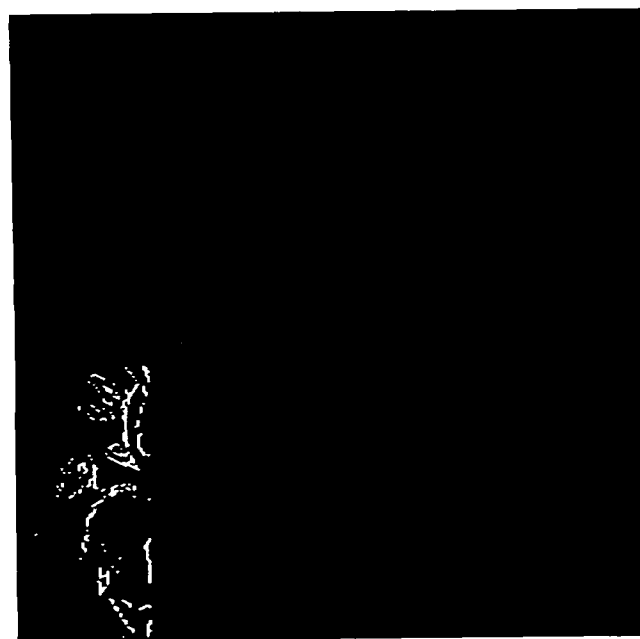


(b) Mapping of image to computation plane

Fig.II.21 Edge Detection Using Simulation of Circular-Elements Sensor (55 Rings, 113 Elements per Ring).

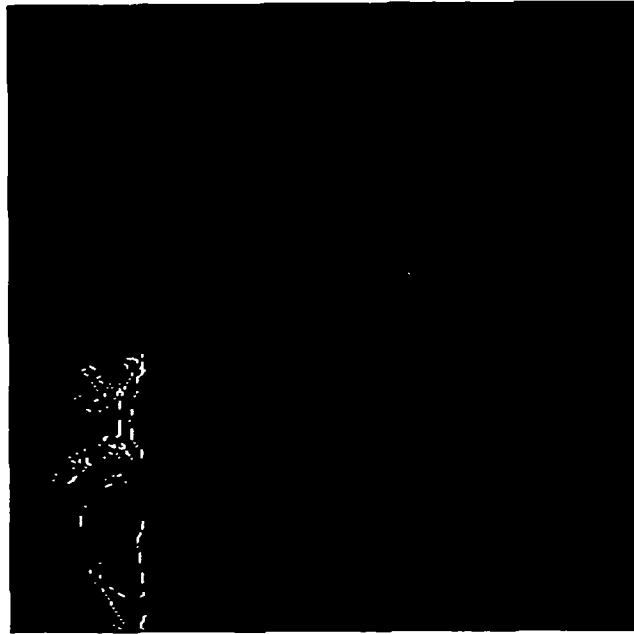


(c) Edge image in CP, absolute value method, threshold = 55



(d) Edge image in CP, Laplacian method, threshold = 15

Fig.II.21 Edge Detection Using Simulation of Circular-Elements Sensor (55 Rings, 113 Elements per Ring).



(e) Edge image in CP, Laplacian-Gaussian, no thresholding

Fig.II.21 Edge Detection Using Simulation of Circular-Elements Sensor (55 Rings, 113 Elements per Ring).

References

- [1] Braccini C., Gambardella G., Sandini G., and Tagliasco V.: A model of the early stages of the human visual system: functional and topological transformations performed in the peripheral visual field, *Biol. Cybern.* 44, 1982, pp. 47-58.
- [2] Braccini C., Gambardella G.: Linear shift-variant filtering for form-invariant processing of linear scaled signals, *Signal Processing* 4, 1982, pp. 209-213.
- [3] Michael R. C.: Retinal processing of visual images, *Scientific America*, May 1969, pp. 104-114.
- [4] Ratliff F.: *Mach Bands: Quantitative studies on neural networks in the retina*, Holden-Day, 1965.
- [5] Hochstein S. and Shapley M. R.: Quantitative analysis of retinal ganglion cell classifications, *J. Physiol.* 262, 1976, pp. 237-264.
- [6] Hochstein S. and Shapley R. M.: Linear and non linear spatial subunits in Y cat retinal ganglion cells, *J. Physiol.* 262, 1976, pp. 265-284.
- [7] Murray I., MacCana F. and Kulikowski J. J.: Contribution of two movement detection mechanisms to central and peripheral vision, *Vis. Res.*, Vol. 23, No. 2, 1983, pp. 151-159.

- [8] Marr D. and Hildreth E.: Theory of edge detection, Proc. R. Soc. Lond. B 207, 1980 ,pp. 187-217.
- [9] Schwartz L. E.: Spatial mapping in the primate sensory projection: analytic structure and relevance to perception, Biol. Cybern. 25, 1977, pp. 181-194.
- [10] Wilson R. H. and Bergen R. J.: A four mechanism model for threshold spatial vision, Vis. Res., Vol. 19, 1979, pp. 19-32.
- [11] Braccini C., Gambardella G., Sandini G. and Tagliasco: Borrowing from the eyes to create robot vision algorithms, Sensor Review, April 1981, pp. 68-72.
- [12] Braccini C., Gambardella G. and Sandini G.: A signal theory approach to the space and frequency variant filtering performed by the human visual system, Signal Processing 3, 1981, pp. 231-240.
- [13] Shanmugam K. S., Dickey M. F. and Green A. J.: An optimal frequency domain filter for edge detection in Digital Pictures, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 1, January 1979, pp. 37-49.
- [14] Marvin E. J. and Ron W. W.: Does the eye contain optimal edge detection mechanisms, IEEE Transaction on Systems, Man, and Cybernetics, Vol. SMC 11, No. 6, June 1981, pp. 441-444.

- [15] Marr D. and Poggio T.: A computational theory of human stereo vision, Proc. R. Soc. Lond. B 204, pp. 301-328.
- [16] Hildreth C. E.: The detection of intensity changes by computer and biological vision systems, Computer Vision, Graphics, and Image Processing 22, 1983, pp. 1-27.
- [17] Levialdi S.: Edge extraction techniques, Fundamentals in computer vision, edited by Faugeras O. D., pp. 117-144.
- [18] Gonzalez C. R., Wintz P.: Digital image processing.
- [19] Pirenne M. H.: Vision and the eye, 2nd, Chapman & Hall, London, 1967.

CHAPTER III

QUALITATIVE MOTION DETECTION ALGORITHMS

III.1. Introduction

In this chapter, algorithms for qualitative detection of motion will be developed. By this we mean that using information from the computation plane (CP), a decision will be made in software on the direction of motion, without a quantitative determination of the increments in the x and y directions in image plane (IP). The following assumptions will be made:

1. The image has been mapped to the computation plane, segmented by edge detection in CP and objects are represented by their edges.
2. Sampling rate is fast enough so that only motions small in comparison to object size need be considered.

The following are cases of interest:

- a. Scaling about the optical axis.
- b. Rotation about the optical axis (OA).
- c. Scaling plus rotation both about the OA.
- d. Translation along a straight line in image plane (IP).
- e. combination of (a) and (d), above.
- f. Combination of (b) and (d), above.
- g. Combination of (a), (b) and (d), above.

These cases have to be considered for single or multiple object scenes with objects either enclosing or non-enclosing the optical axis. Initially we will consider only single object scenes.

III.2 Magnification and Rotation

III.2.1. Object Edges Enclosing the Optical Axis.

We will refer, first, to the case of edges enclosing the optical axis (in IP, obviously). Cases (a) and (b) are then very simple because for these two situations the properties of the log conformal transformation produce the well known invariances with shifting along the " μ " and " ν " axes of the CP plane, for scaling and rotation, respectively. In these two cases, in addition to easy determination of the direction of motion, it is also easy to determine corresponding points between the original and the displaced edges, because edges remain invariant in shape and are only shifted. Local maxima and minima, easy to determine in the original and the new (from now on the "previous" and the "current" images, PI and CI, respectively) will be corresponding points in both images. An example of a scaled and a rotated image in CP for the random figure in IP of Fig. 1 are given in Figs. 1(a) and 1(b), respectively. For the scaled case, if the difference image (DI), i.e., the subtraction of the PI from the CI, contains CI points, scaling has been up ($M > 1$); if the DI contains PI points, $M < 1$, Figs. 2(a) and 2(b), respectively. In other words, the object is closer to the camera or farther away, respectively. For the rotation case, the DI contains both CI and PI points, but maxima and minima values of μ do not vary. We wish to determine whether rotation is CW or CCW. Realizing that the vertical axis value in

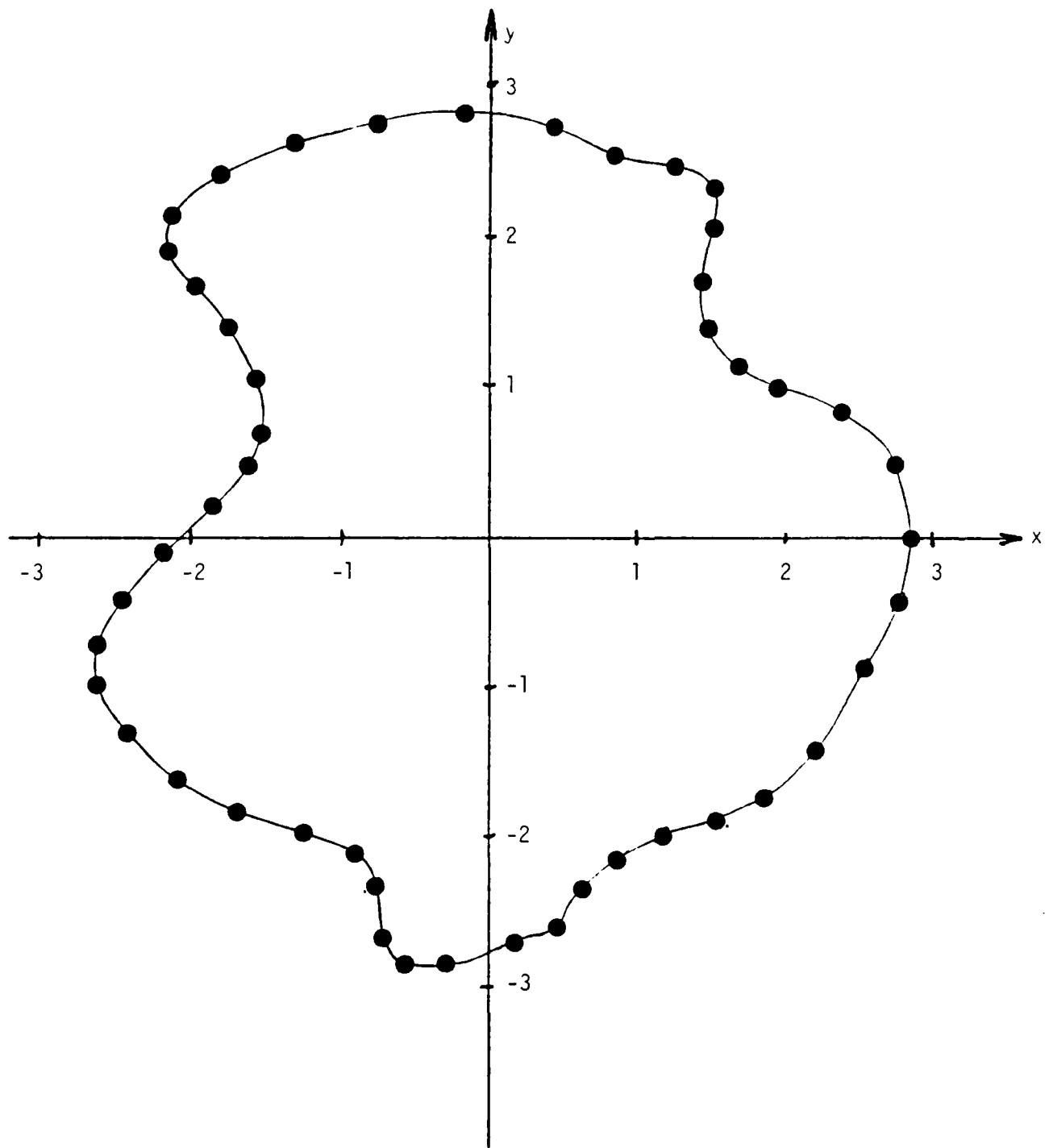


Fig III.1 Random figure in image plane

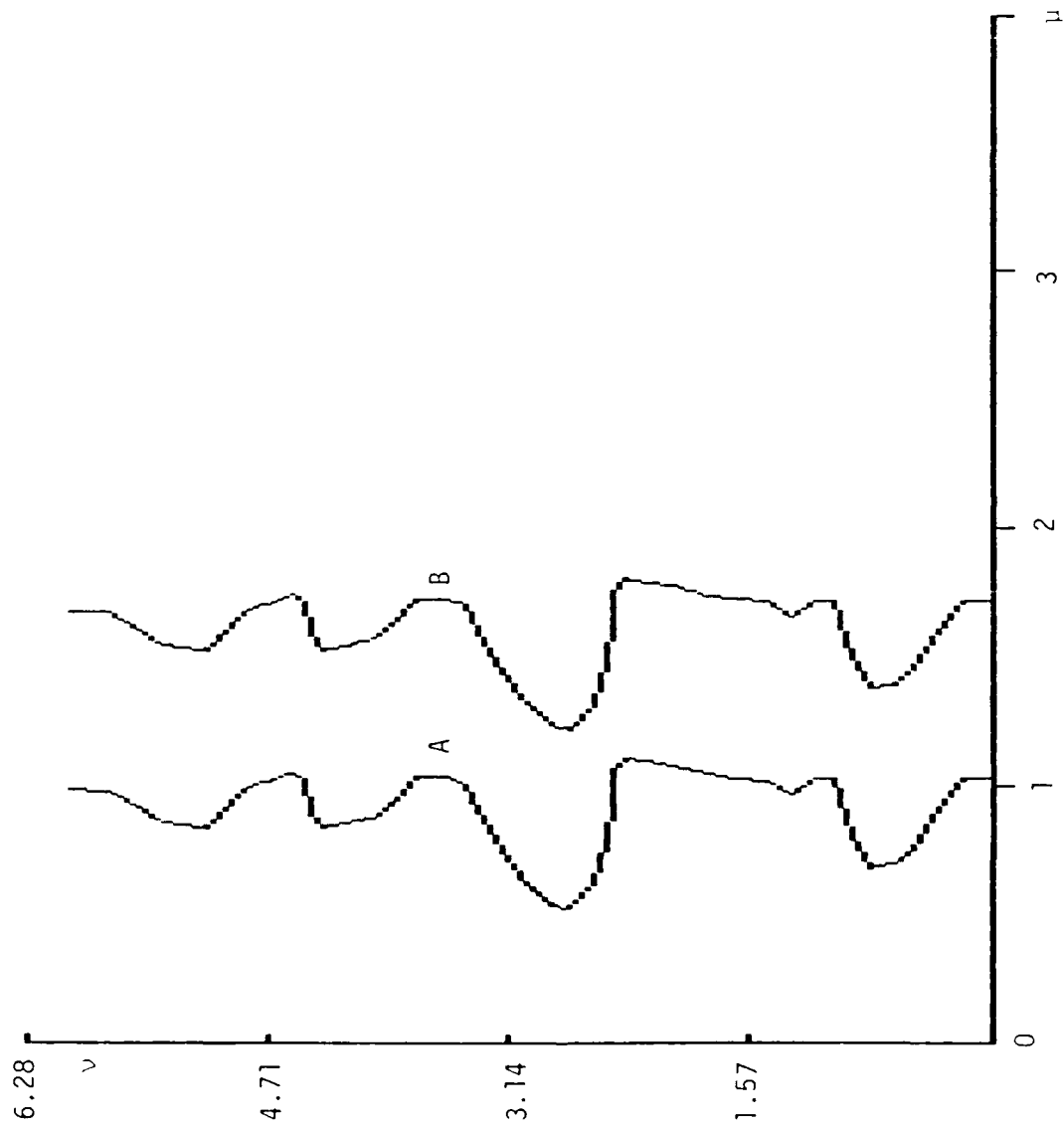


Fig III.1(a) Random figure edge in computation plane

(A) Original; (B) Figure magnified by $m=2$ about optical axis in IP

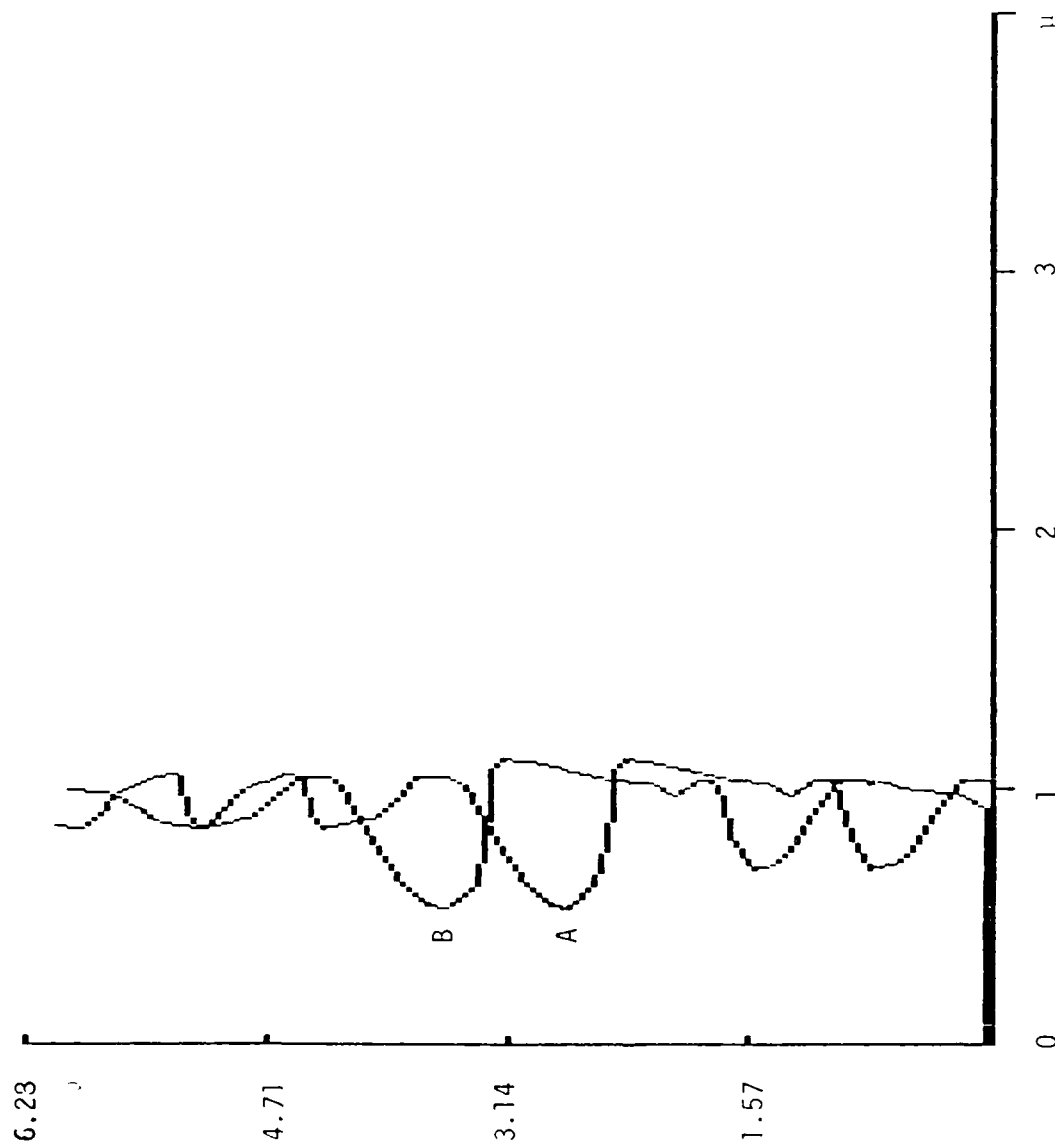


Fig III.1(b) Random figure edge in computation plane

(A) Original; (B) Figure rotated by $+45^\circ$ about optical axis in IP

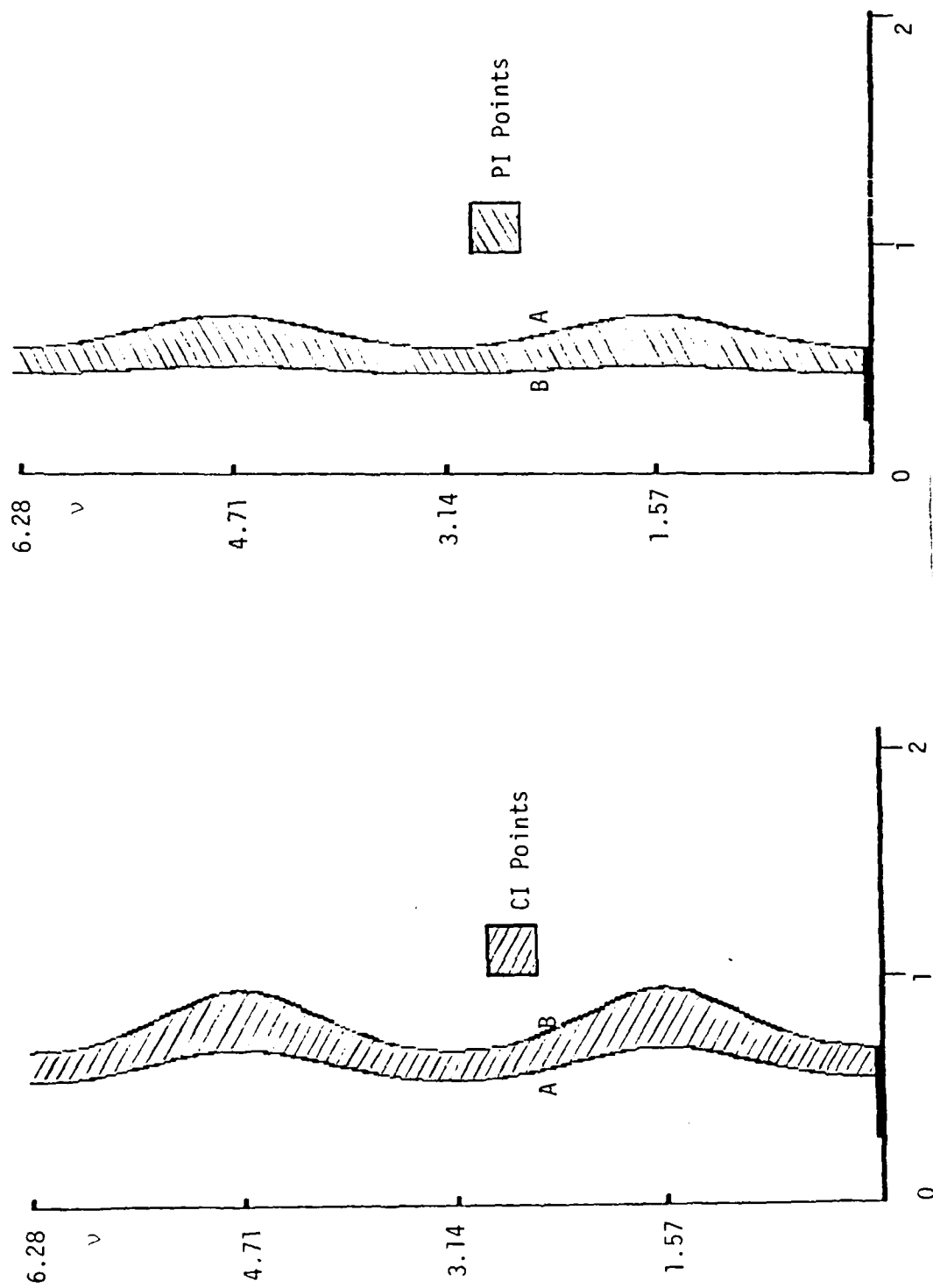


Fig III.2 Ellipse: (a) Original, A and magnified by 2.3, B; (b) Original, A and scaled by 0.8, B.

the CP is equal to the phase angle of the corresponding point in the IP, it is enough to locate the value of ν corresponding to either the maximum or the minimum value of μ in both PI and CI. If this corresponding value of ν is larger in CI, it means that the rotation angle is positive, i.e., CCW, Fig. 3(a). If this corresponding value of ν is smaller in CI, the rotation is negative, i.e., CW, Fig. 3(b). For scaling plus rotation, a combination of the two properties described above applies. For test, locate either the maximum or the minimum value of μ for both CI and PI. Shift the CI edge up or down by the difference in ν value between maxima and minima of μ . This will eliminate rotation and, if only scaling and rotation have occurred, the DI will now contain only CI points or PI points. Figure 4 shows an original image plus a scaled/rotated one. Notice that the shape of the edge has not varied, it has just shifted in both the μ and the ν directions. For an image that has been translated, the DI image will contain both PI and CI points, Fig. 5, and in addition, maxima and minima values of μ will be different in both images.

III.2.2 Objects with Edges not Enclosing the Optical Axis

It is convenient to emphasize that the properties pertaining to scaling and rotation apply regardless of whether the objects enclose the optical axis or not, as long as the scaling or the rotation are with respect to the optical axis. If the original test figure is translated until it does not enclose the optical axis and then it is either scaled or rotated or both, the properties stated earlier still apply. i.e., object shape remains invariant, but there will be translation along the μ axis for scaling and along the ν axis for rotation. There is, however, a fundamental difference between the edge of an object

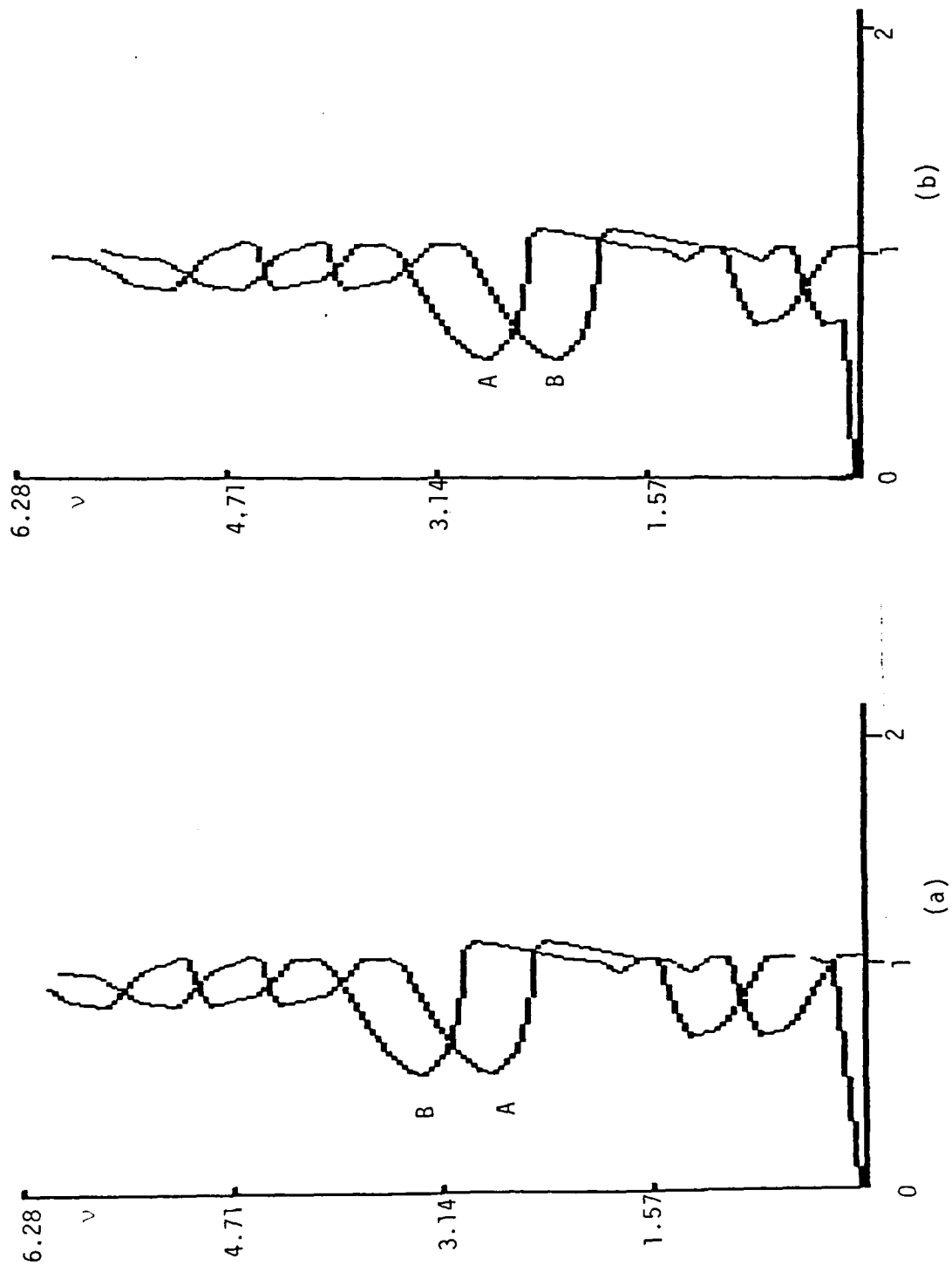


Fig III.3 Random figure: (a) Original, A and rotated by $+30^\circ$, B; (b) Original, A and rotated by -30° , B.

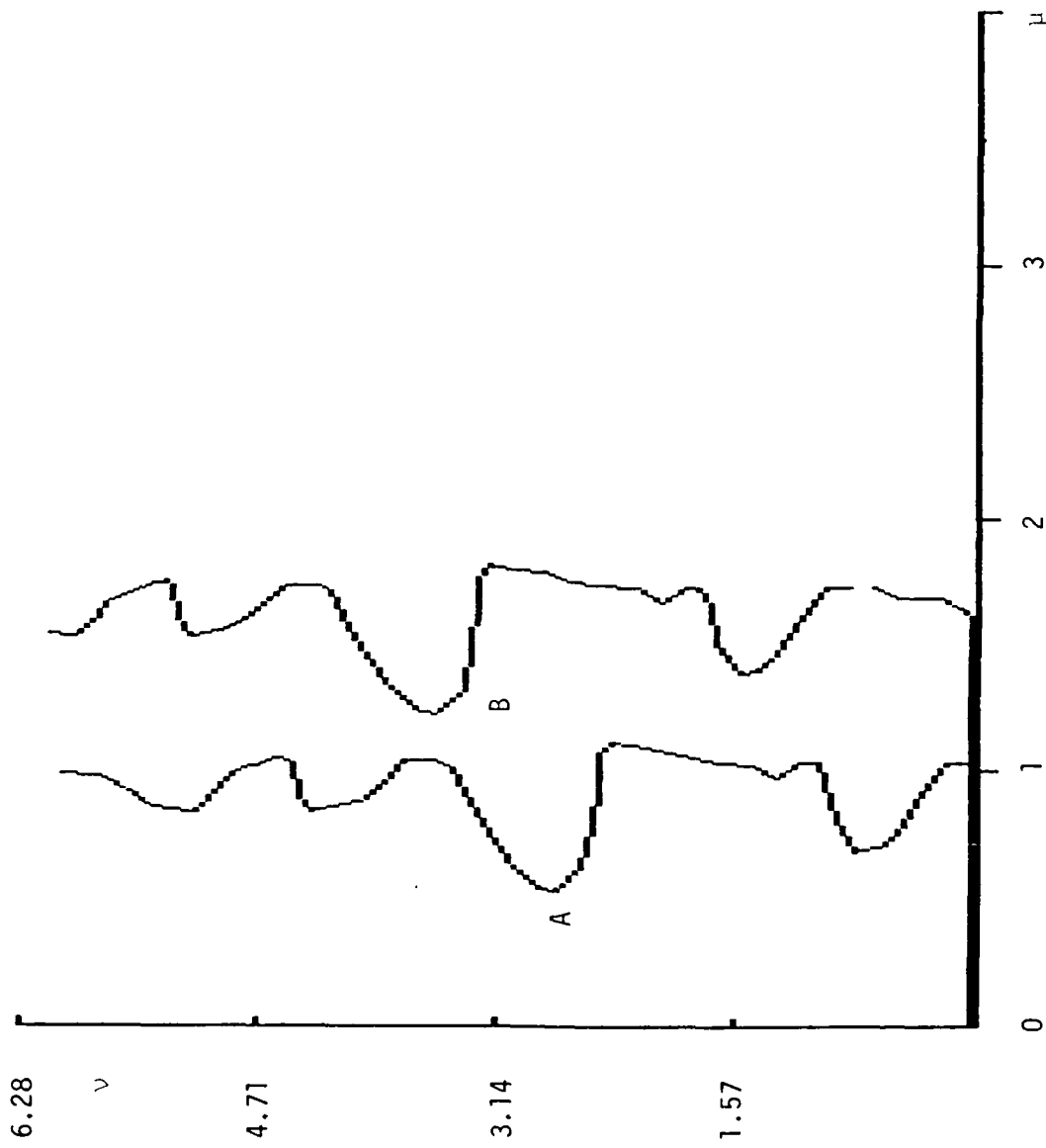


Fig III.4 Random figure edge in CP: (A) Original; (B) Magnified by $m=2$ and rotated by $\alpha=+45^\circ$ both about optical axis

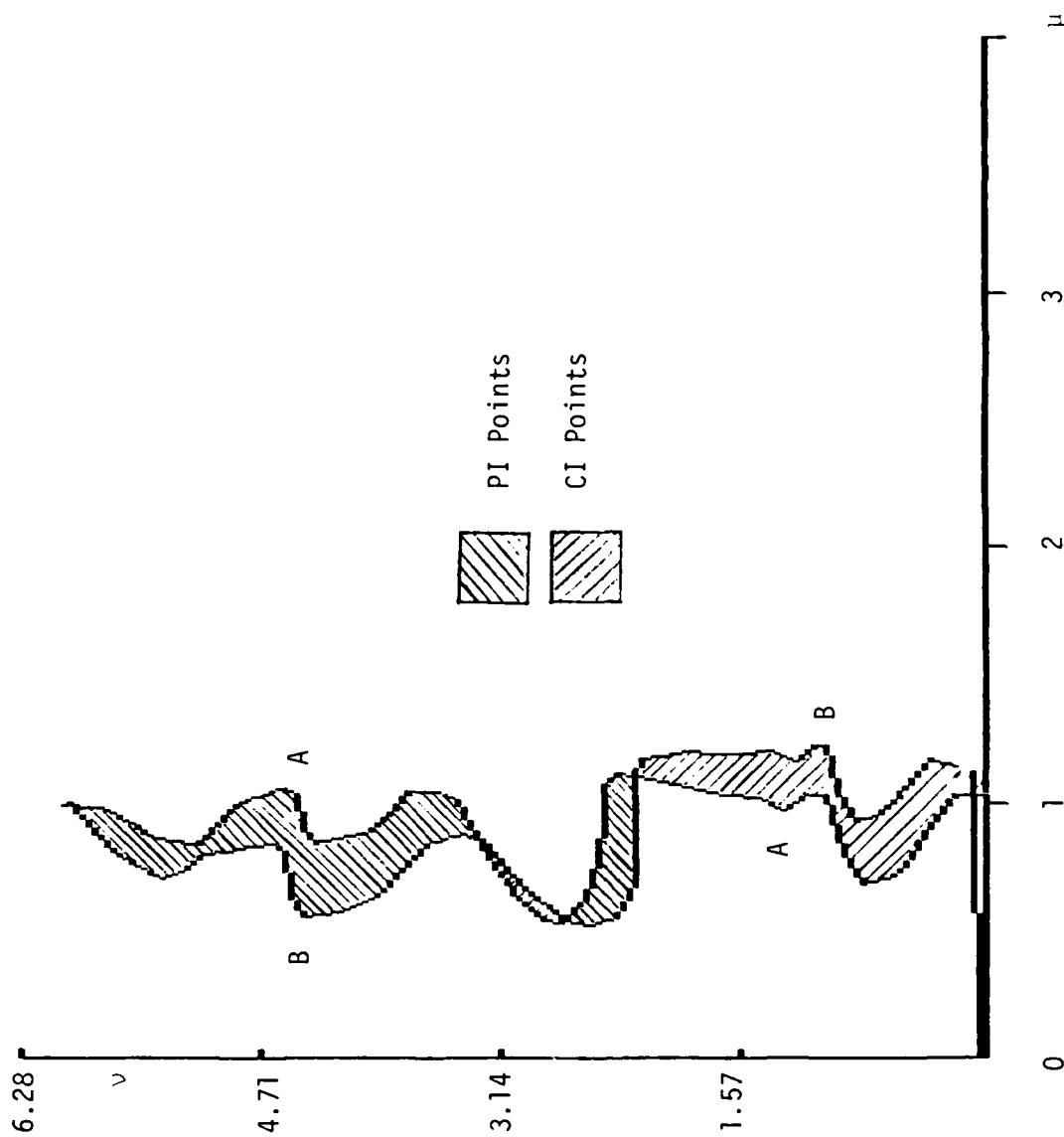


Fig. III.5 Random figure edge in CP: (A) Original; (B) Translated by $\mu=0.3$ in IP.

enclosing the optical axis and one that does not. In the former case, as the edge contour is followed in IP from an initial point until that point is reached again, the phase angle (value of ν in the CP) varies by 2π . This means that, in CP, the object's edge is open. On the other hand, for an object not enclosing the optical axis, when the contour is followed in IP, the net variation of the angle is zero and the edge is closed also in CP, see Fig. 6. The object in this figure is the same as the original for Fig. 1, but it has been displaced until it does not enclose the optical axis. Notice that for three different sizes of the object ($M=0.8$, $M=1$, and $M=1.2$), the shape and size in CP do not vary, but the shifting along the x axis does, Fig. 7. Notice also that, contrary to what happens when the object encloses the optical axis, the difference image contains both PI and CI points. This means that a different criterion is needed for both cases in order to determine direction of motion. Notice that scaling of noncentered objects in IP maps to CP in a form similar to translation in IP and that consequently, a criterion similar to that used by Jain [1] can be applied to obtain motion information. For scaling of off-center objects about the optical axis, compute the difference image. If only scaling exists, there will be a left hand side (LHS) and a right hand side (RHS) edge composed of PI and CI only, respectively, or vice versa in the DI. If the LHS edge has CI points and the RHS edge has PI points, scaling has been $M < 1$. If the LHS edge has PI points and the RHS edge has CI points, scaling has been $M > 1$, refer to Figs. 6 and 8, respectively.

Notice that in either case, minimum and maximum values of ν remain invariant, which can be used as a criterion to determine whether only scaling

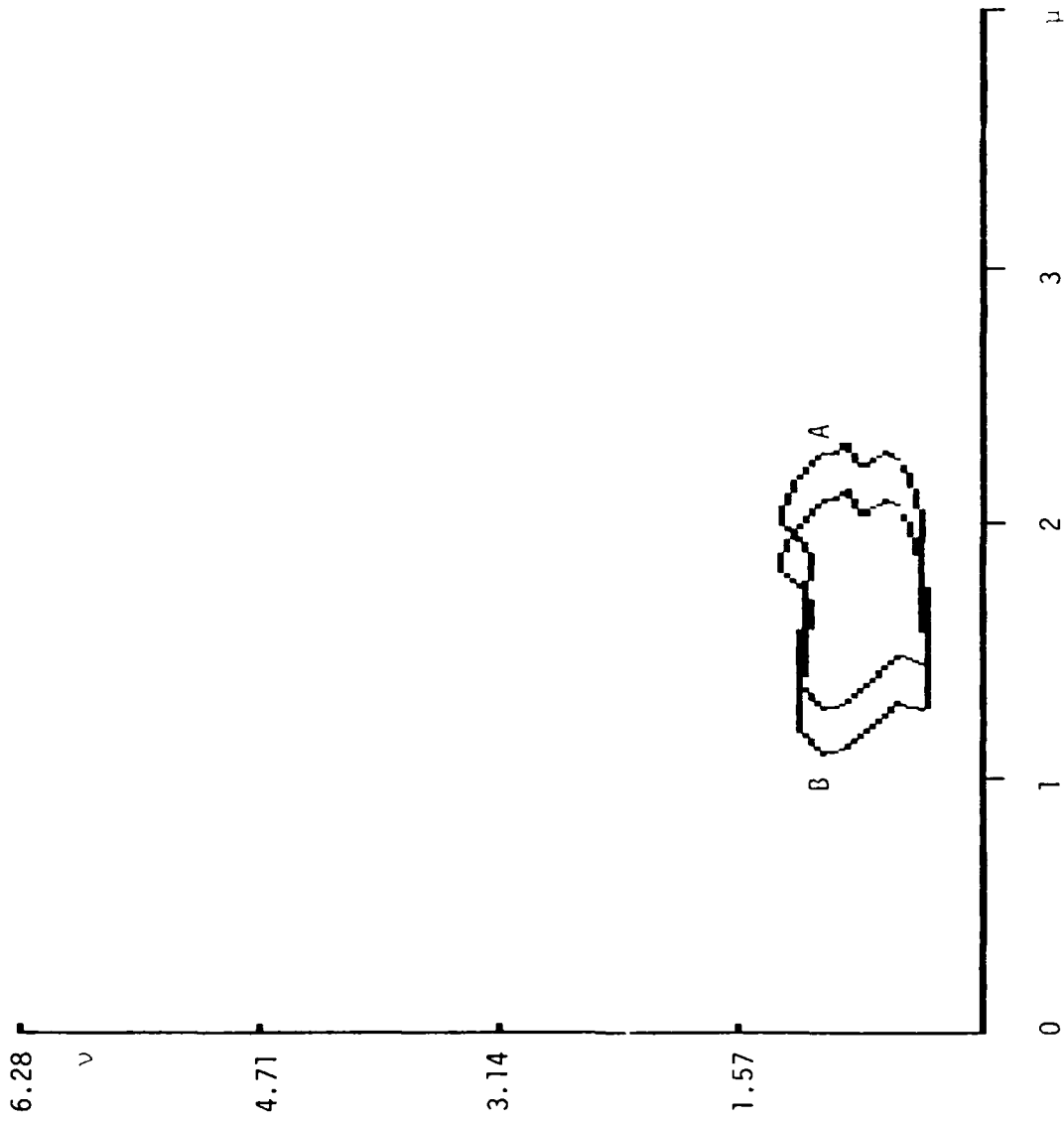


Fig III.6 Mapping of off-center, by (4,4) units, random figure.
(A) $m=1$; (B) $m=0.8$

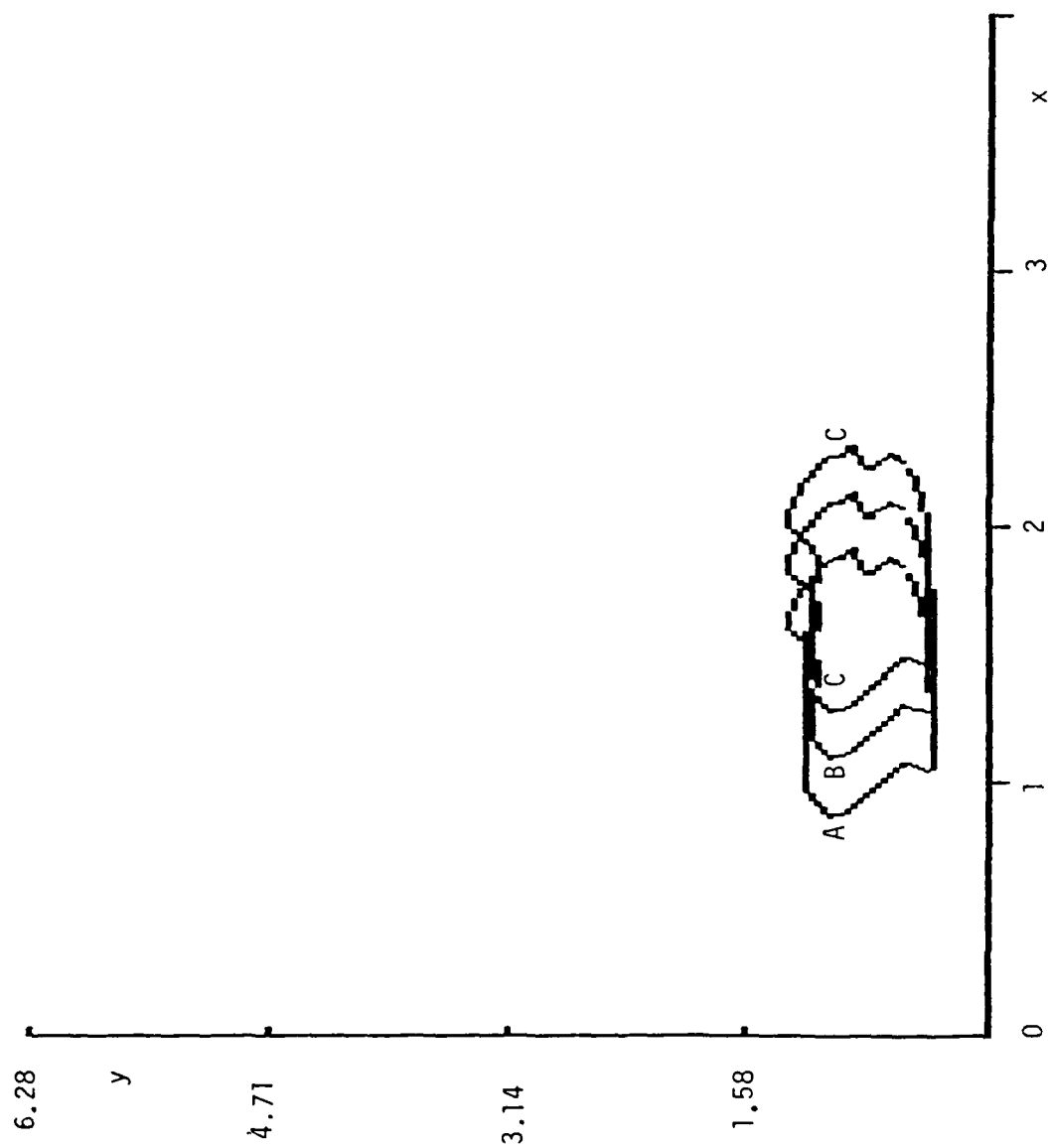


Fig III.7 Mapping of off-center, by (4,4) units, random figure.
(A) $m=0.8$, (B) $m=1$, (C) $m=1.2$

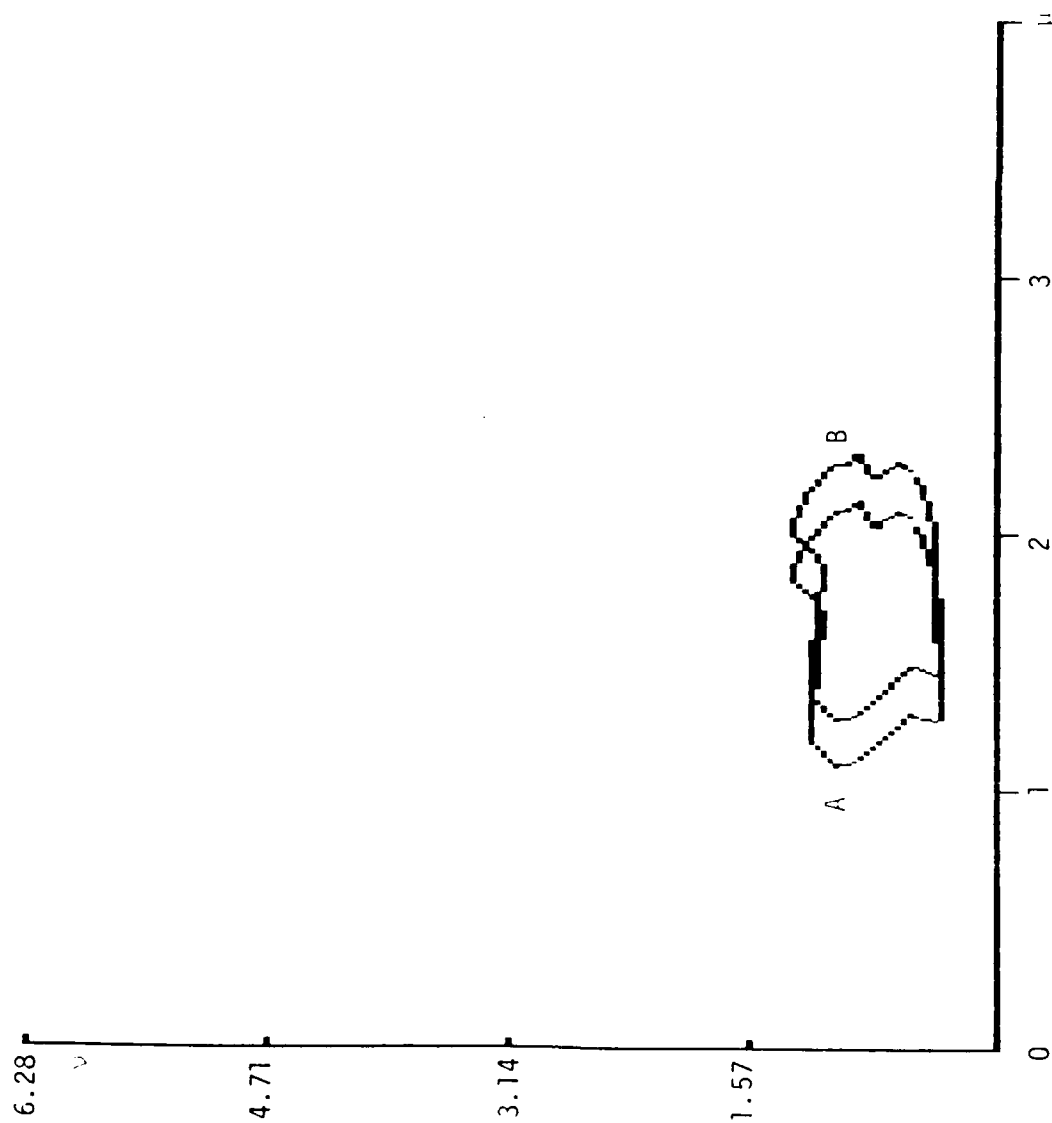


Fig III.8 Mapping of off-center, by (4,4) units, random figure.
(A) $m=1$; (B) $m=1.2$

exists.

For rotation, the off-center case is also different from the centered case. The closed edge in CP moves up or down, respectively, for CCW and CW rotation without changing either shape or size. The difference image will contain both CI and PI points, Fig. 9. If only rotation occurs, the DI will contain an upper and a lower edge with only PI and CI points, respectively, or vice versa. If the upper DI edge contains only CI points and the lower edge only PI points, rotation has been CCW. If the contrary is true, rotation has been CW.

Notice that in either case, minimum and maximum values of μ remain invariant, which can be used as a criterion to determine whether only rotation exists.

For the scaling plus rotation case, the method described for the "centered" case, also applies to the off-centered case. If this method is applied to Figs. 10 and 11, correct results are obtained. Of course, once the two images are "aligned" with respect to the y axis, the criterion for off-center scaled images must be applied.

We have, thus, covered cases (a), (b) and (c) for single objects, centered and off-center.

III.2.3. Multiple Objects

For multiple objects, the same properties and criteria apply that apply for single objects. Two general cases can occur: (a) motion of the camera; (b) motion of objects in the scene.

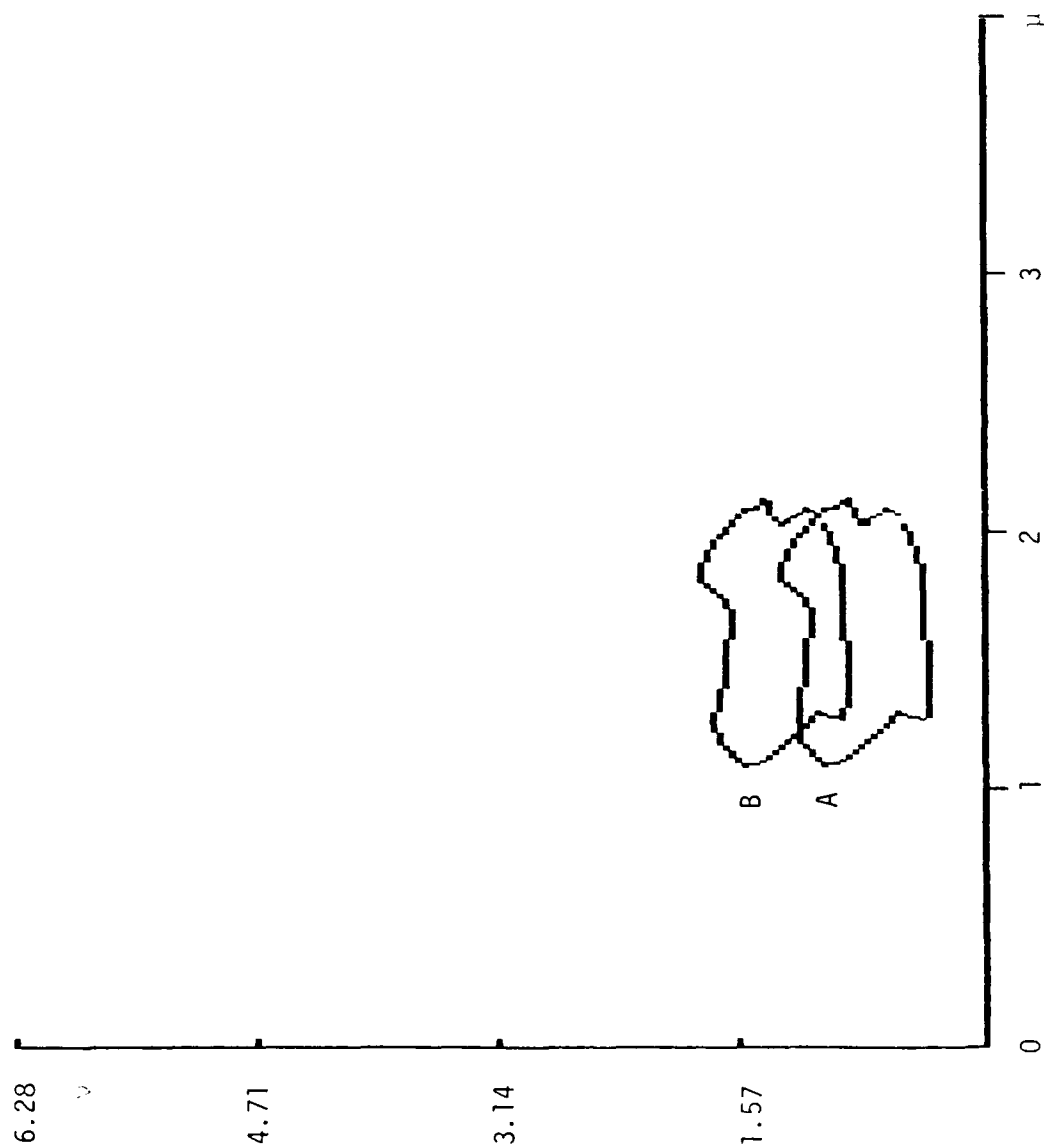


Fig III.9 Mapping of off-center, by (4,4) units, random figure.

(A) $m=1$, $\alpha=0$; (B) $m=1$, $\alpha=+30^\circ$

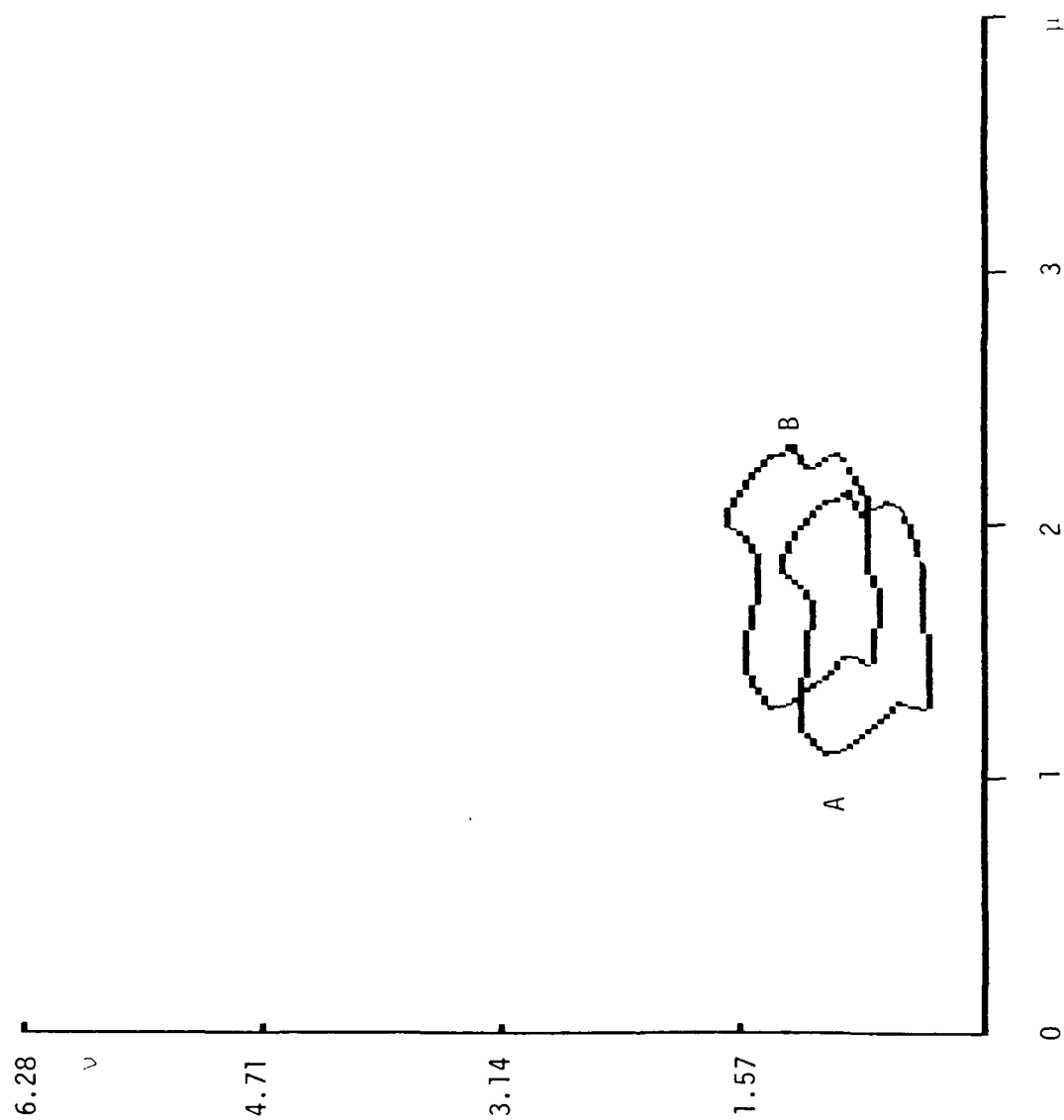


Fig III.10 Mapping of off-center, by (4,4) units, random figure.
(A) $m=1$, $\alpha=0$; (B) $m=1.2$, $\alpha=20^\circ$

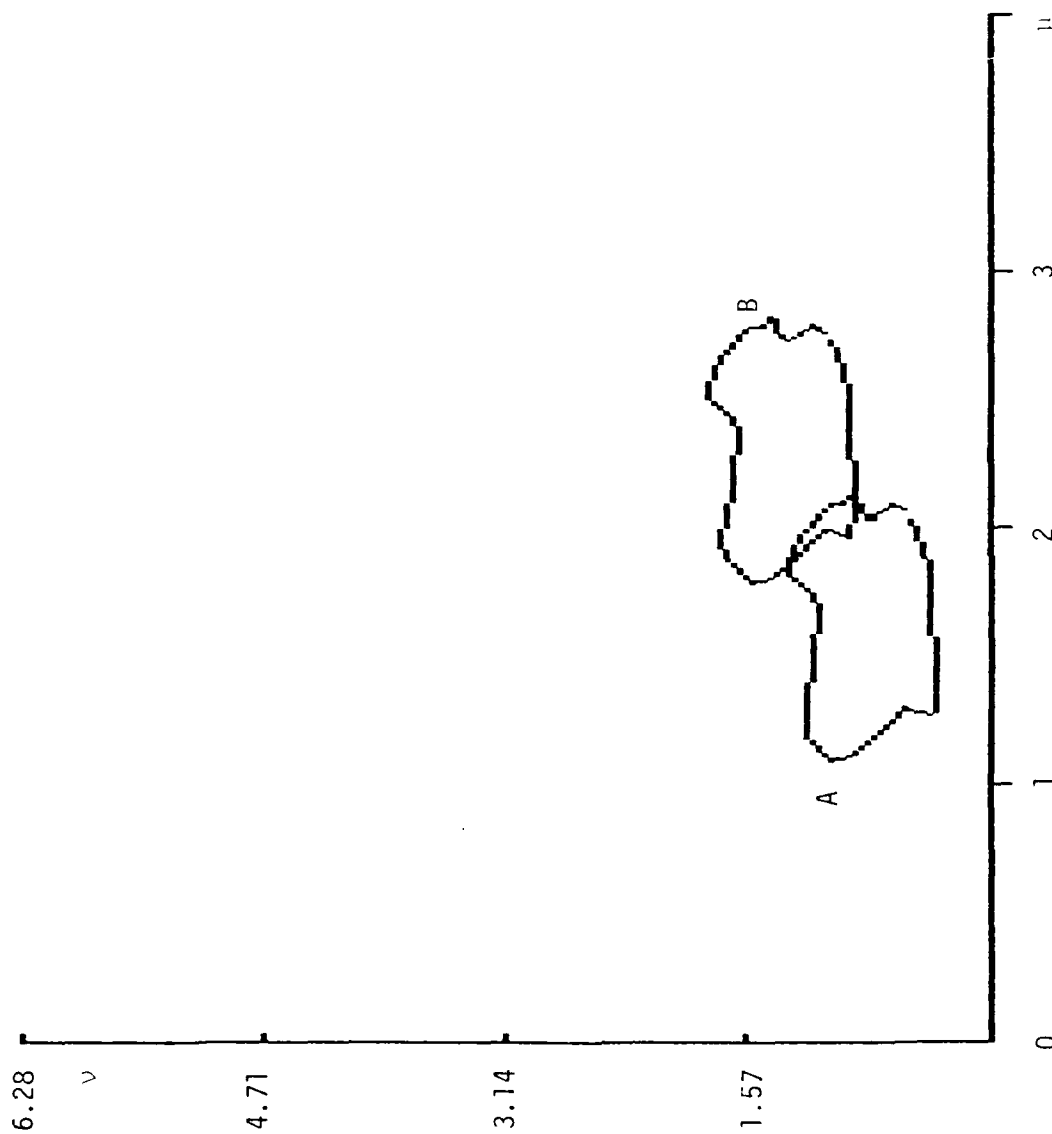


Fig III.11 Mapping of off-center, by (4,4) units, random figure.

(A) $m=1$, $\alpha=0$; (B) $m=2$, $\alpha=30^\circ$

For case (a), if the objects are static, the complete scene will vary simultaneously. In other words, if the orientation of the camera does not vary but the camera-scene distance does, all the objects will be scaled. For rotation about the optical axis, all objects will rotate simultaneously. Some objects will be centered and some off-center. The centered square and off-center rectangle of Fig. 12(a) are mapped as indicated in Fig. 12(b). Figure 12(c) shows the mapping of the scene after rotation by 15° and Fig. 12(d) after scaling by $M=1.2$. Simultaneous rotation and scaling are shown in Fig. 13. Notice that although scaling or rotation alone produce, in this case, overlapping PI and CI for the off-center object, the combination of both produces nonoverlapping images, Fig. 13.

III.3 Translation Along a Straight Line in Image Plane

This is a more difficult problem than scaling and rotation, because the mapping from IP to CP does not possess, in this case, the useful properties that apply to scaling and rotation.

For geometric figures such as squares, rectangles, circles and ellipses, as well as for most nongeometrical objects, the following qualitative algorithm applies for centered figures:

Consider the DI, i.e., the labeled absolute value of the difference between the current image and the previous image intensities.

1. Horizontal motion

a) If the DI contains more CI points for values $-\pi/2 < \nu < \pi/2$ than for $\pi/2 < \nu < 3\pi/2$ and more PI points for $\pi/2 < \nu < 3\pi/2$ than for $-\pi/2 < \nu < \pi/2$,

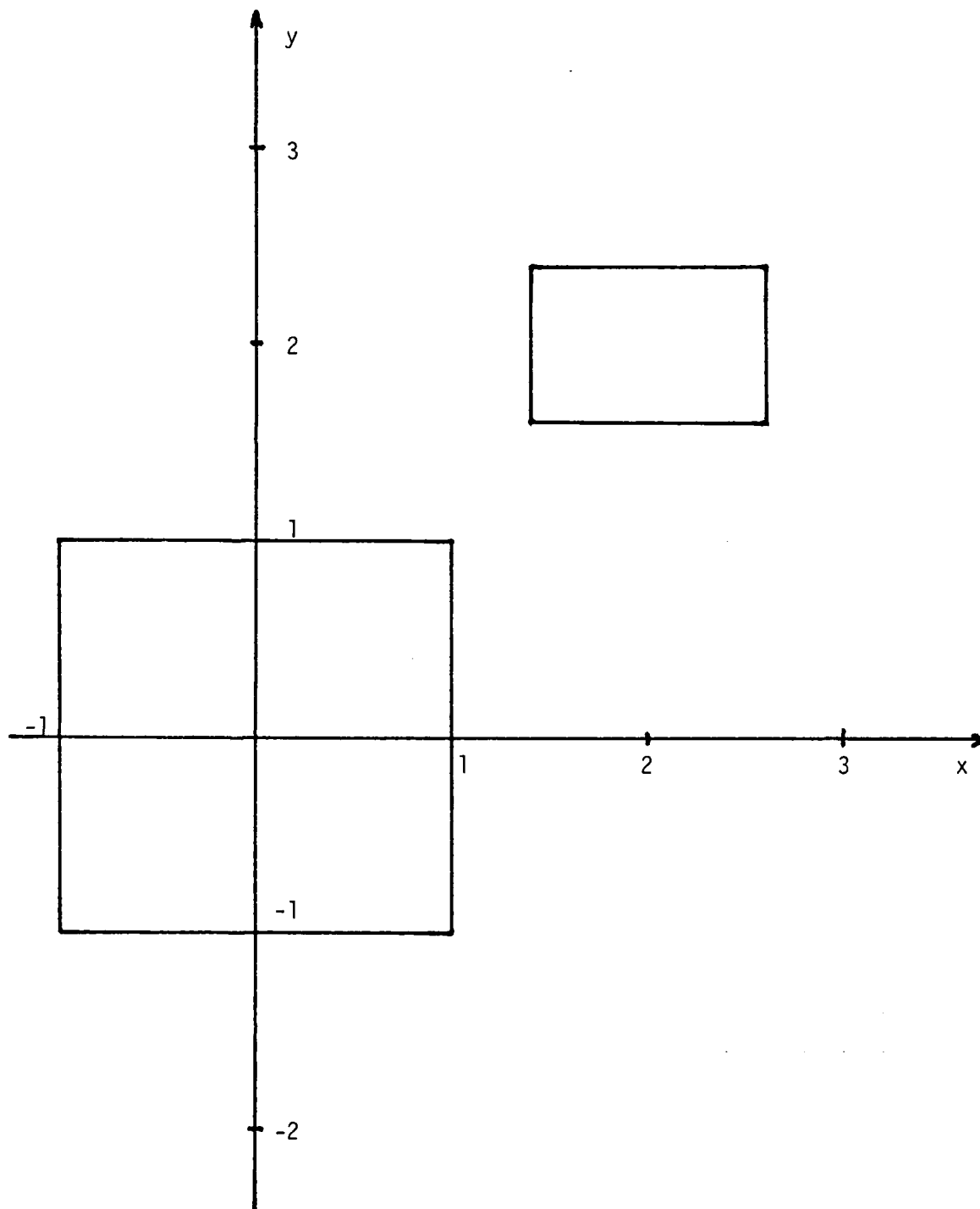


Fig III.12(a) Centered square and off-center rectangle in image plane

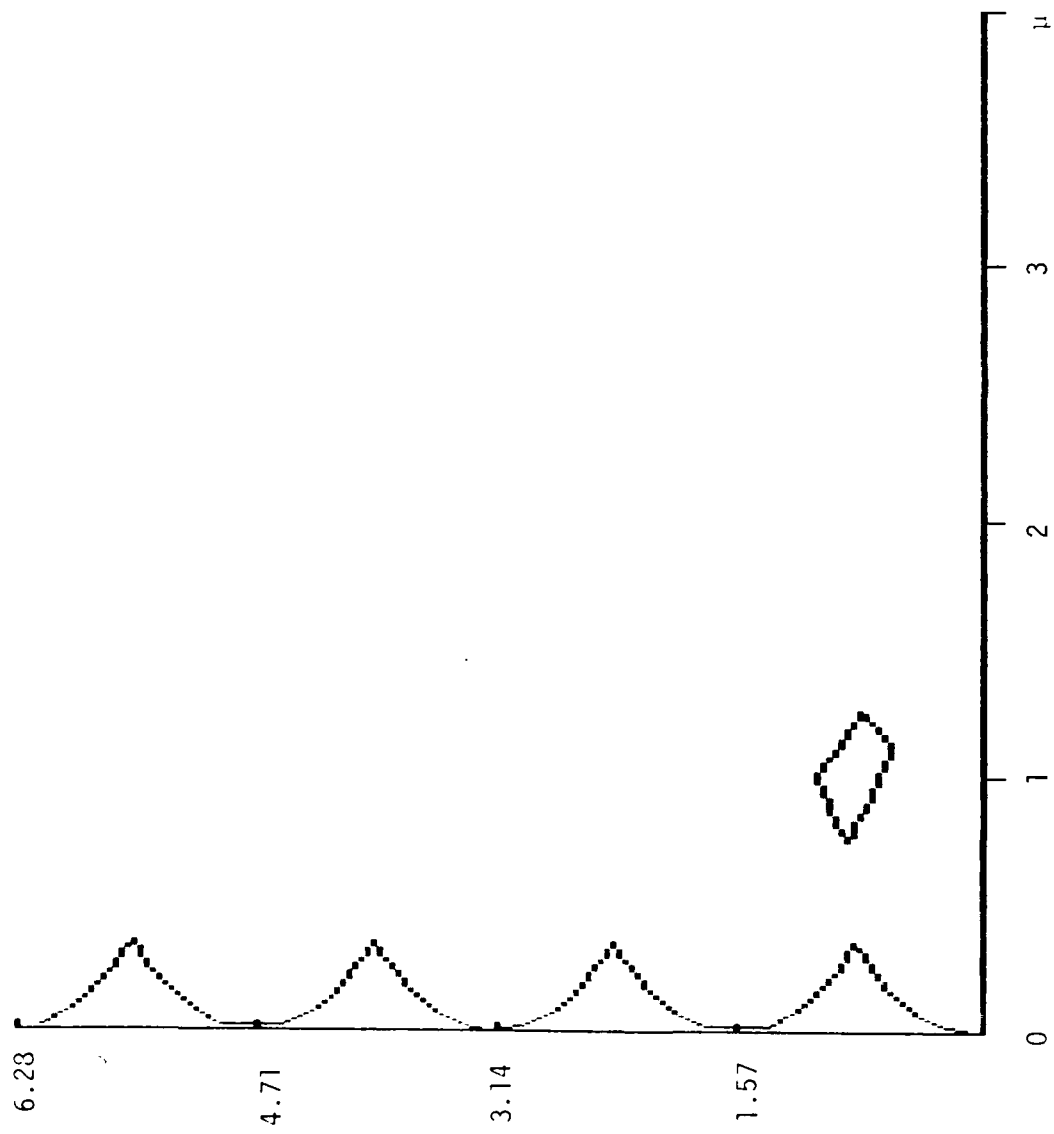


Fig III.12(b) Mapping of the two objects in Fig. 12(a) to the CP.

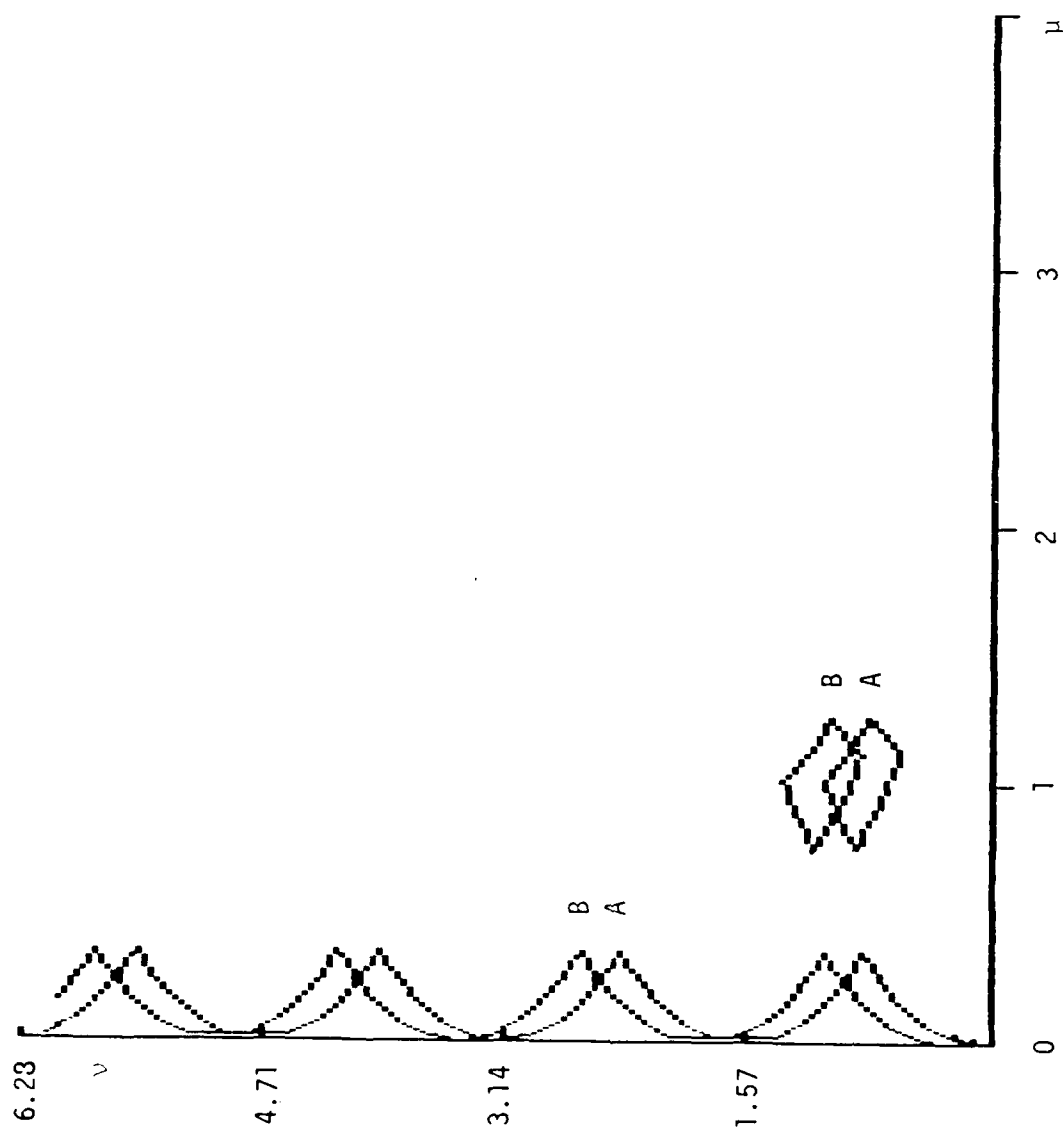


Fig III.12(c) Mapping of Fig. 12(a) to CP.
(A) Original; (B) $m=1$, $\alpha=+15^\circ$

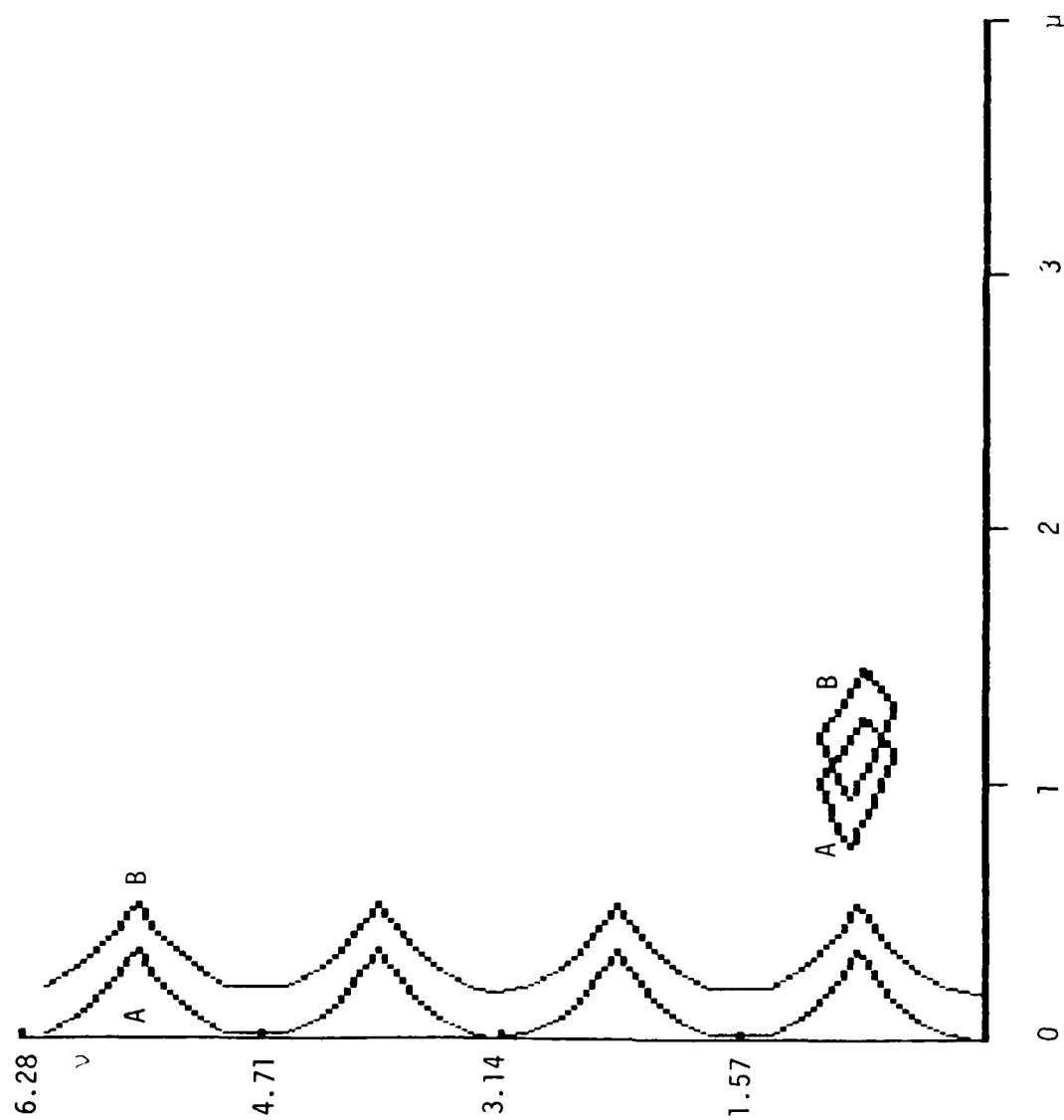


Fig III.12(d) Mapping of Fig. 12(a) to CP.

(A) Original; (B) $m=1.2$, $\alpha=0$

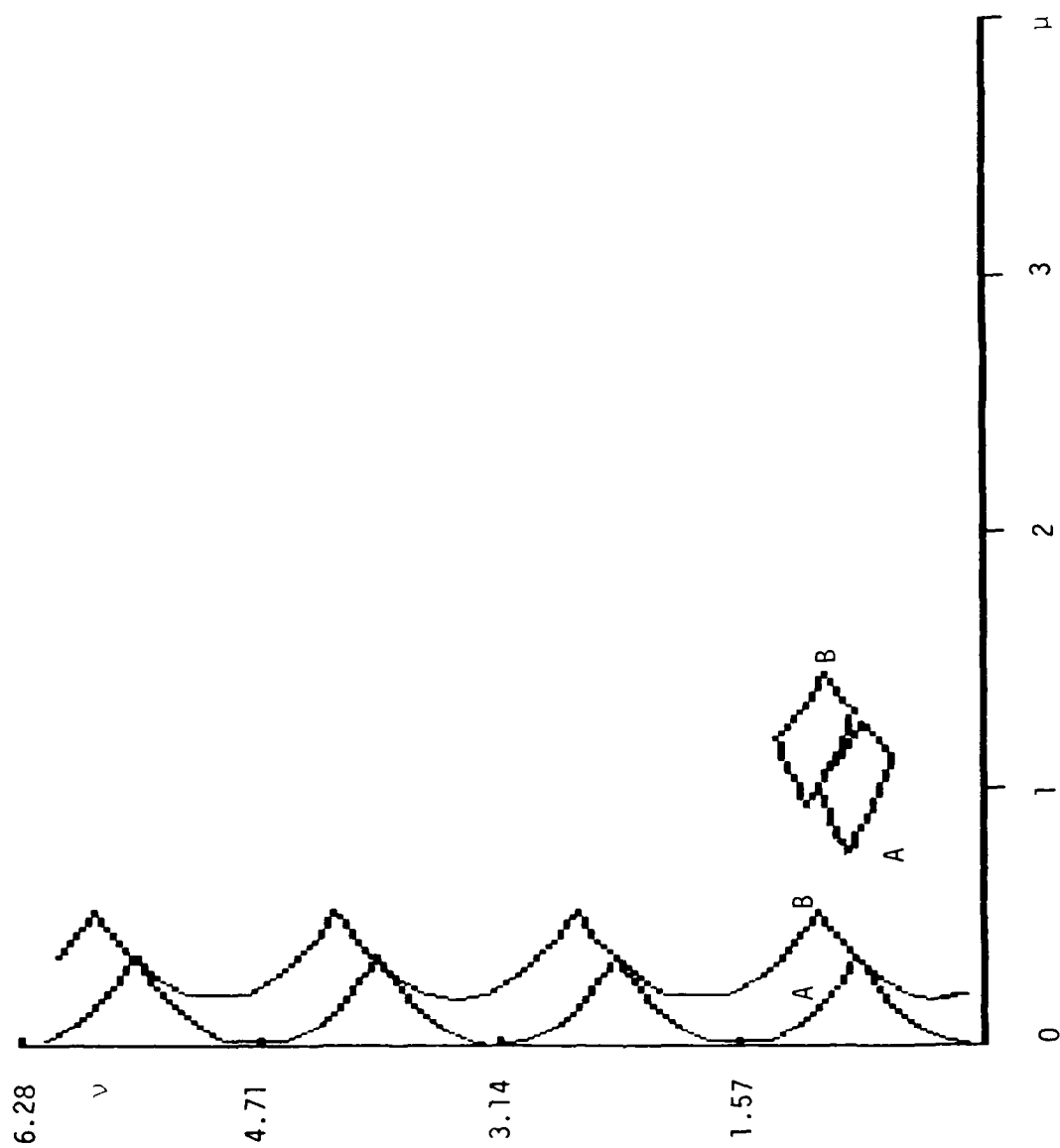


Fig III.13 Mapping of Fig. 12(a) to CP.
(A) Original; (B) $m=1.2$, $\alpha=45^\circ$

motion is the +x direction in IP.

b) If the DI contains more CI points for values $\pi/2 < \nu < 3\pi/2$ than for $-\pi/2 < \nu < \pi/2$, and more PI points for $-\pi/2 < \nu < \pi/2$ than for $\pi/2 < \nu < 3\pi/2$ motion is in the -x direction in IP.

2. Vertical Motion

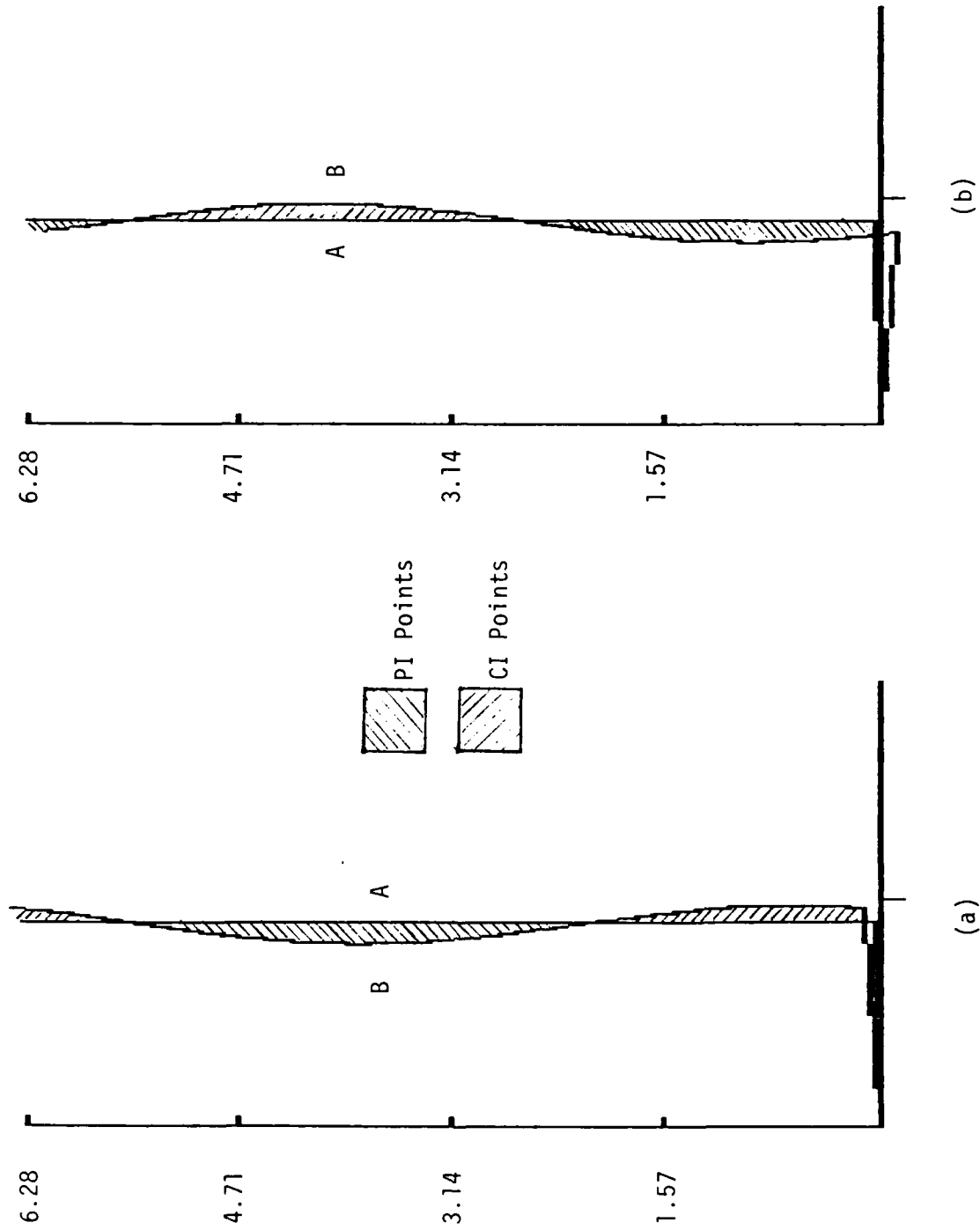
a) If the DI contains more CI points for $0 < \nu < \pi$ than for $\pi < \nu < 2\pi$ and more PI points for $\pi < \nu < 2\pi$ than for $0 < \nu < \pi$, motion is in the +y direction in IP.

b) If the DI contains more CI points for $\pi < \nu < 2\pi$ than for $0 < \nu < \pi$ and more PI points for $0 < \nu < \pi$ than for $\pi < \nu < 2\pi$, motion is in the -y direction in IP.

Examples for geometric objects are given in Figs. 14 to 16.

The method also applies to random figures such as the one shown in Fig. 1 which is a difficult figure because of its concave-convex configuration. Figures 17 to 22 represent the results of translations in different directions, all of which are correctly determined by the above method.

For noncentered figures, there is a change in figure shape, in addition to shifting in the computation plane, for translation in the image plane. The qualitative algorithm described above for centered figures does not apply in this case. The problem is under study at present. Some general properties of the natural log conformal transformation can be used. For example, if motion is along a straight line at a certain angle, although the noncentered object will diminish in size in CP as it moves away from the origin in IP, the "v" value in CP (corresponding to the angle between the line of motion



(a)

(b)

Fig III.14 Circle with $r=6$; edge in CP.

(a) (A) Centered; (B) off-center by (0.8, 0.6) in IP

(b) (A) Centered; (B) off-center by (-0.6, -0.8) in IP

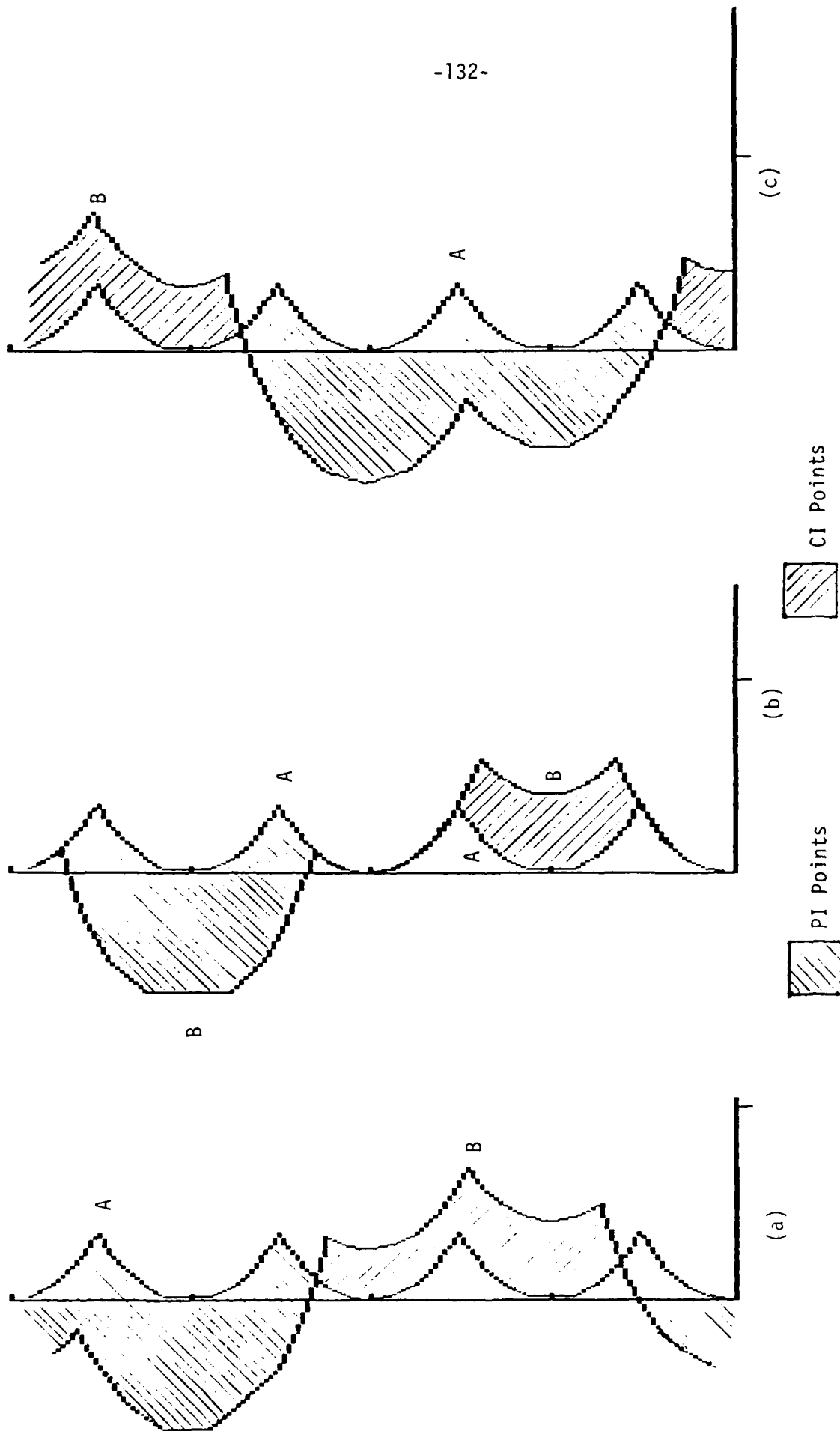


Fig III.15 Square with side $\ell=2$; edge in CP.

- (a) (A) Centered, (B) off-center by $(-0.4, 0.5)$ in IP
- (b) (A) Centered, (B) off-center by $(0, 0.5)$ in IP
- (c) (A) Centered, (B) off-center by $(0.5, -0.4)$ in IP

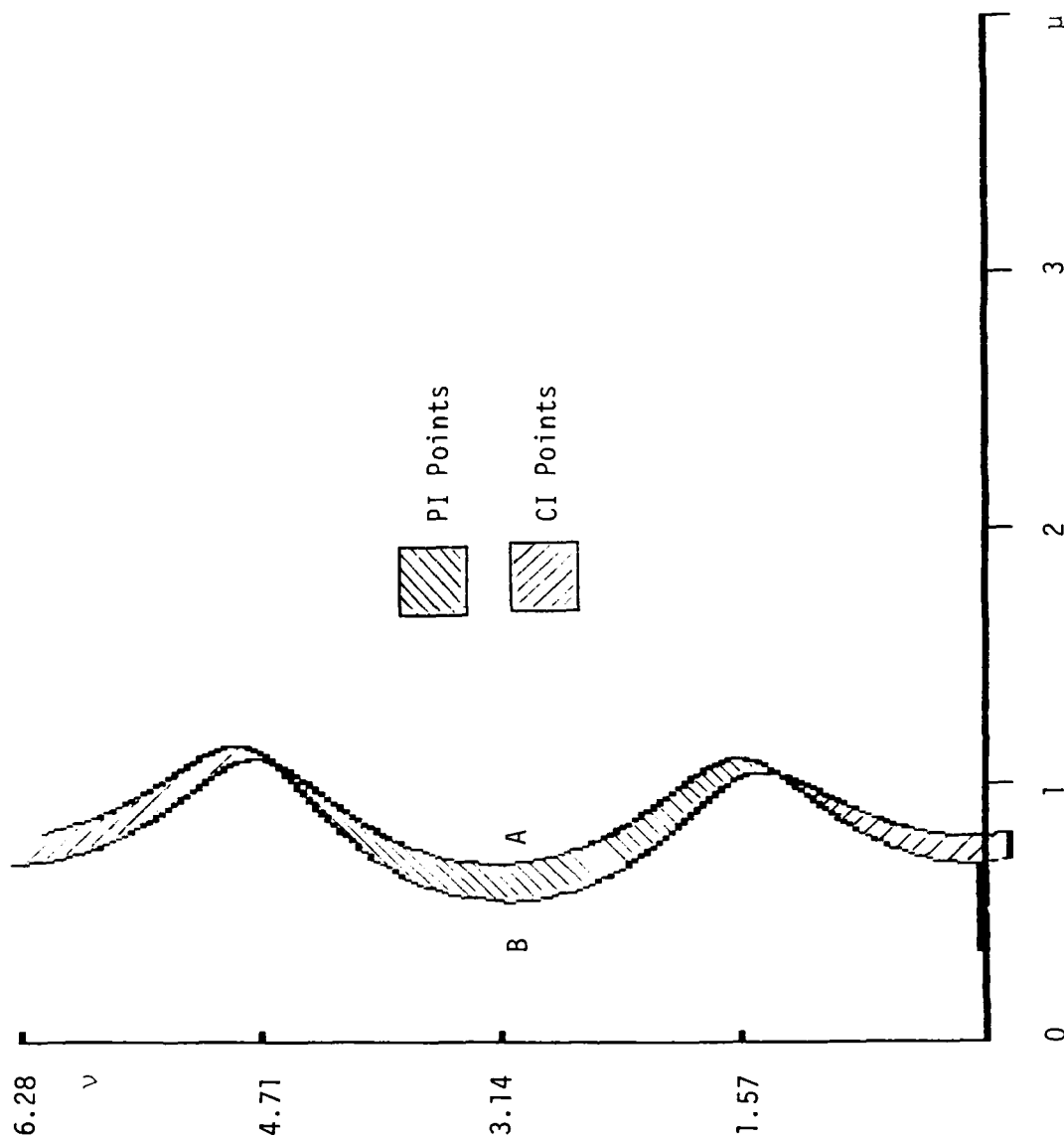


Fig III.16 Ellipse with semiaxis $a=4$, $b=3$; edge in CP.
(a) Centered; (b) off-center by $(1,-1)$ in IP

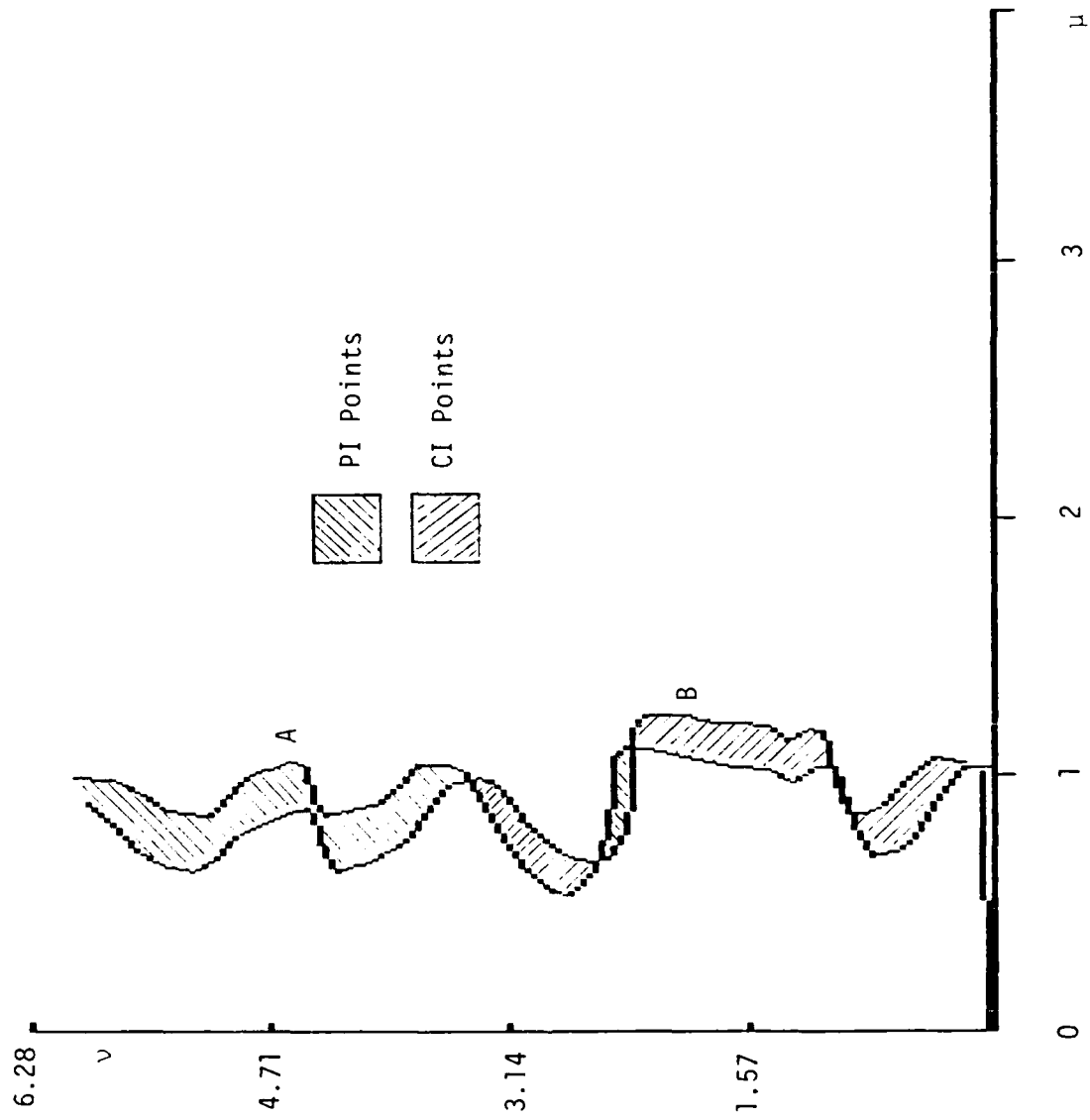


Fig III.17 Random figure edge in CP.
(a) Centered; (b) Translated by (0, 0.5) in IP

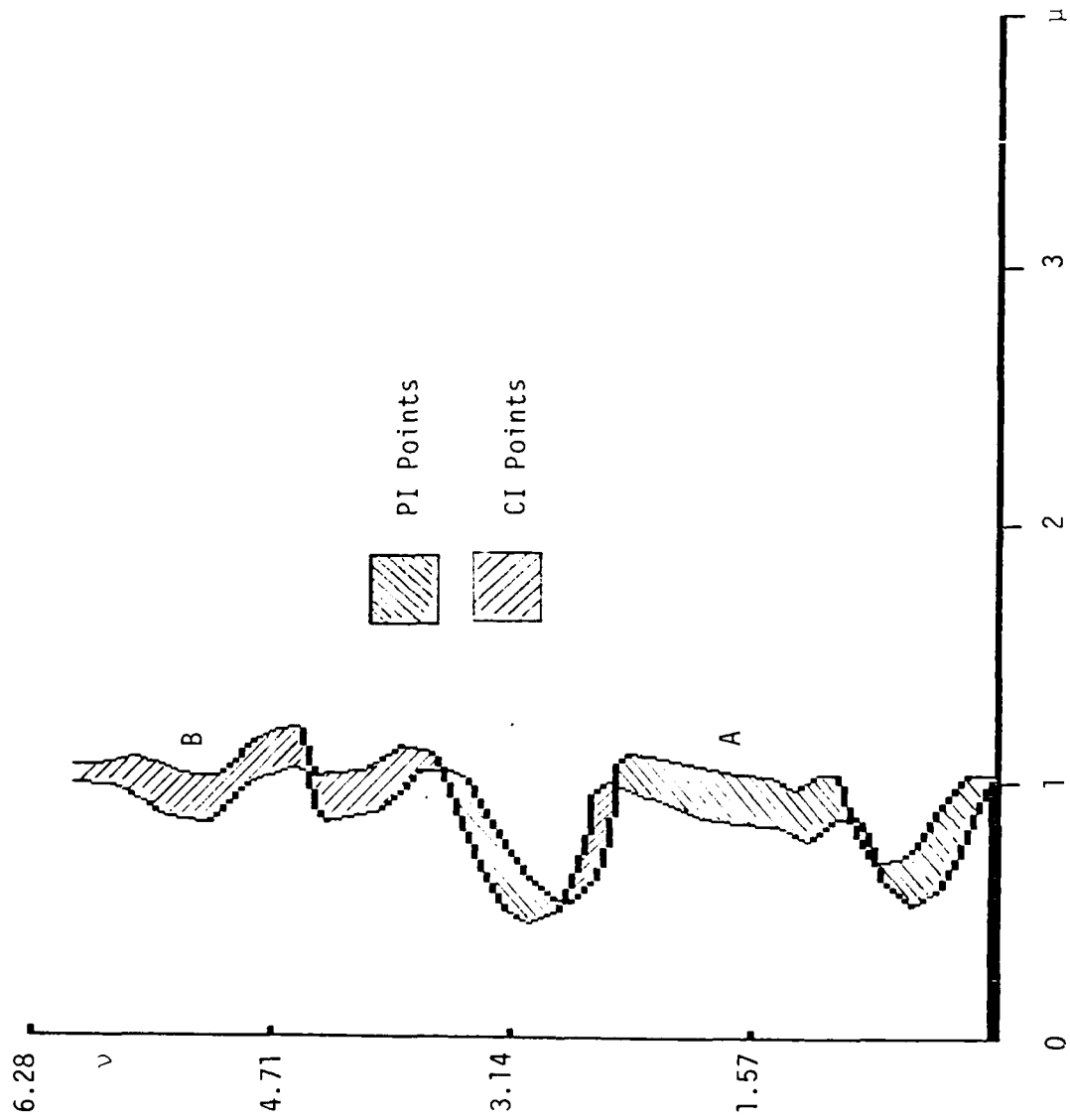


Fig III.18 Random figure edge in CP.
(a) Centered; (b) Translated by (0, -0.5) in IP

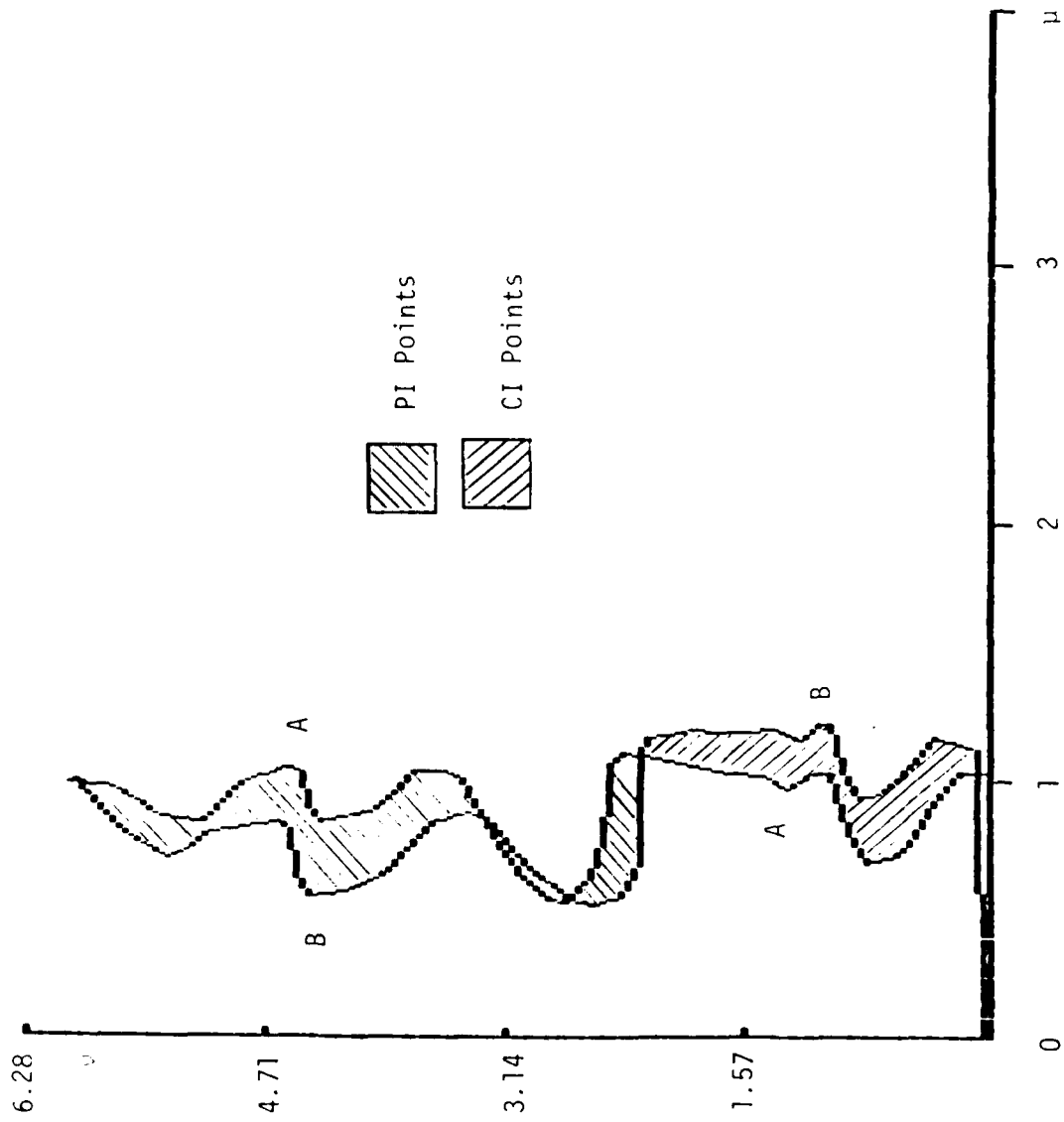


Fig III.19 Random figure edge in CP.
(a) Centered; (b) Translated by (0.3, 0.5) in IP

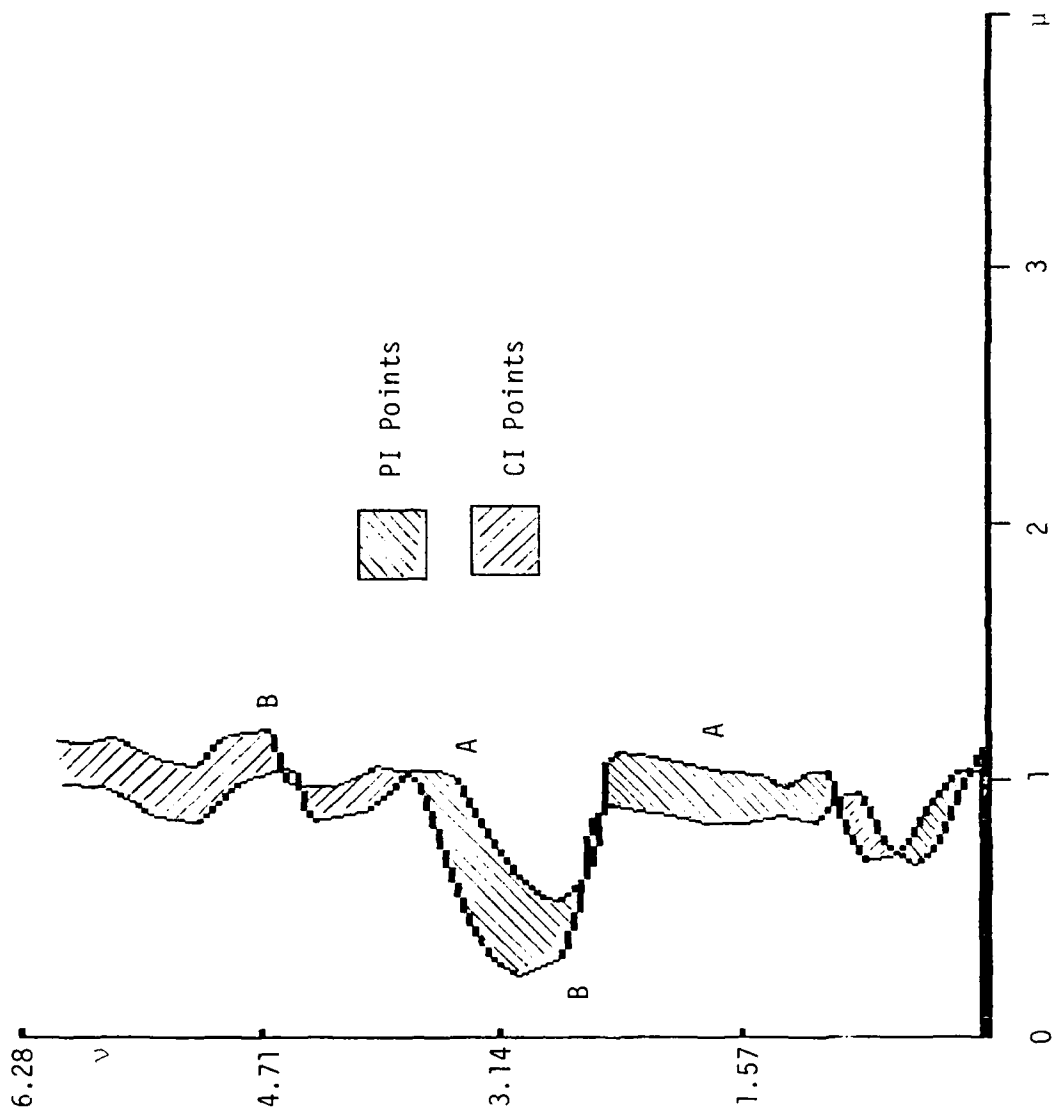


Fig III.20 Random figure edge in CP.
(a) Centered; (b) Translated by (0.3, -0.5) in IP

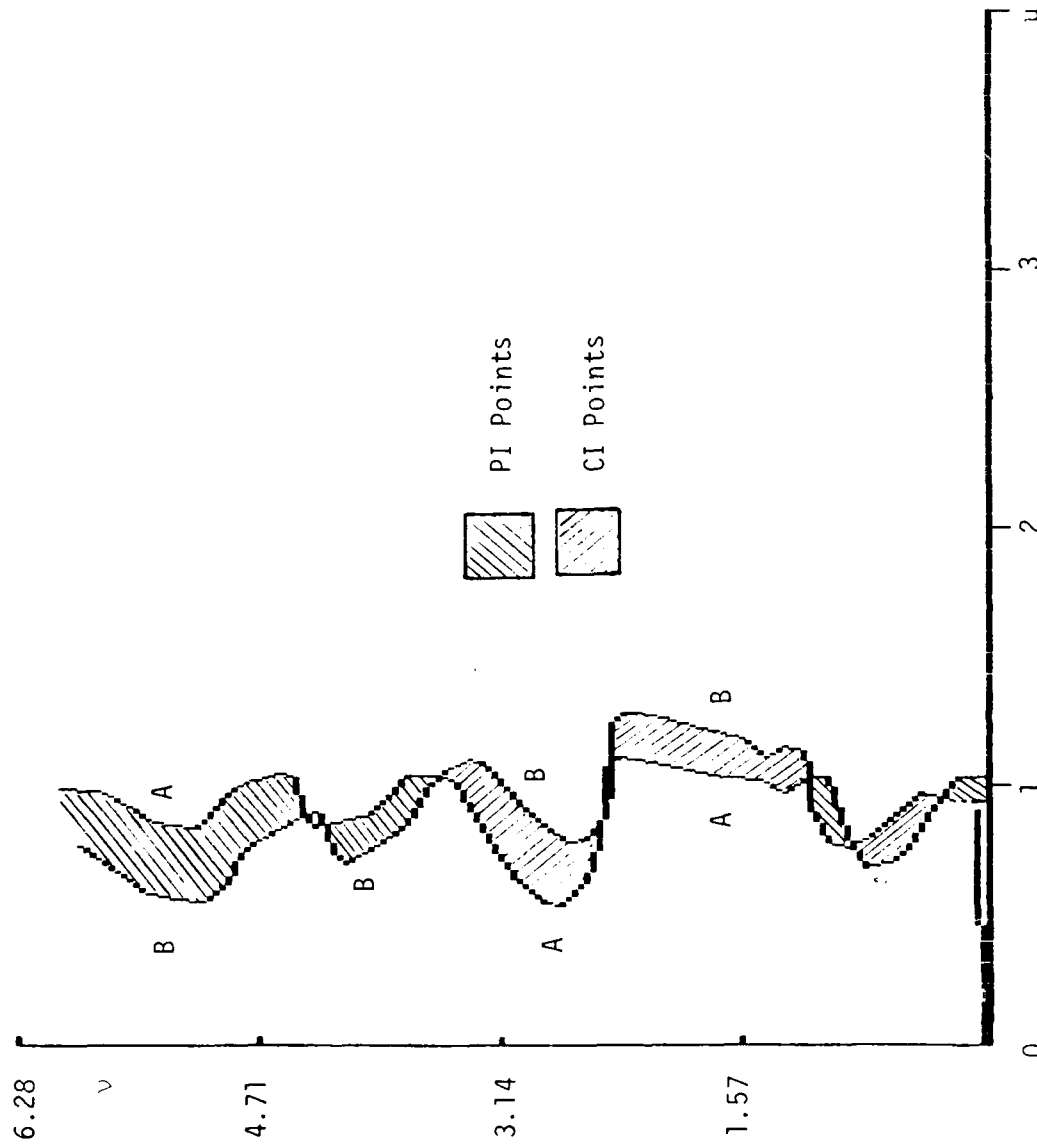


Fig III.21 Random figure edge in CP.
(a) Centered; (b) Translated by $(-0.3, 0.5)$ in IP

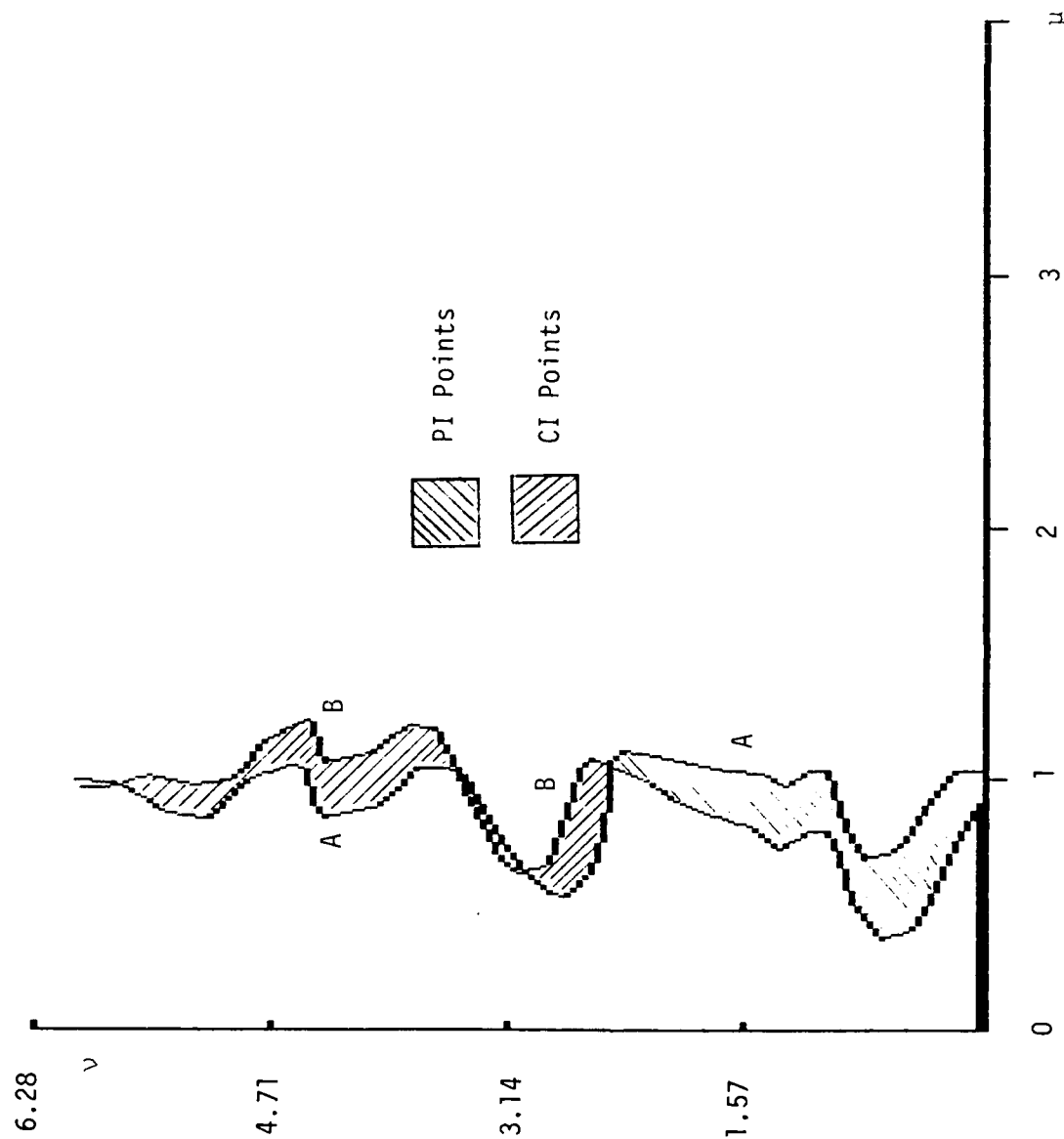


Fig III.22 Random figure edge in CP.

(a) Centered; (b) Translated by $(-0.3, -0.5)$ in IP

and the horizontal axis in IP) will remain constant for a given feature of the object's edge. When the random figure of previous examples is off-center by (4,4) units, the line from the "center" of the figure to the origin is at 45 degrees. For off-center at (5,5), the figure is moving along the line at 45 degrees in IP and the displacement has been (1,1). Figure 23 represents the situation in CP. Comparing this figure with Fig. 6 which, we recall, represents the random figure off-center by (4,4) with $m=1$ and with $m=0.8$, we see that the CP edge images are relatively similar. There is, however, a very fundamental difference: the area of the two figures in Fig. 6 is exactly the same, while in Fig. 23 it is different. A simple way of distinguishing between scaling (or rotation) and translation is by comparing the area of the PI and the CI objects of interest. Figure 24 illustrates the properties of translation along straight lines for off-center figures. A and B are the PI and CI for motion along a straight line at 60 degrees; B is displaced by (1,1.73) with respect to A. C and D are the PI and CI for motion along a straight line at 120 degrees; D is displaced with respect to C by (-1,1.73).

III. 4 Conclusions and Future Plans

In this chapter we have presented qualitative algorithms which permit to determine the direction of motion for scenes under magnification and rotation and, for centered figures, for translation in IP. These algorithms can be applied to multiple object images for nonelastic motion (i.e., all objects are magnified, rotated or moved at the same rate). When completed, these algorithms could be used in conjunction with the quantitative algorithms presented in the next chapter for target or object tracking.

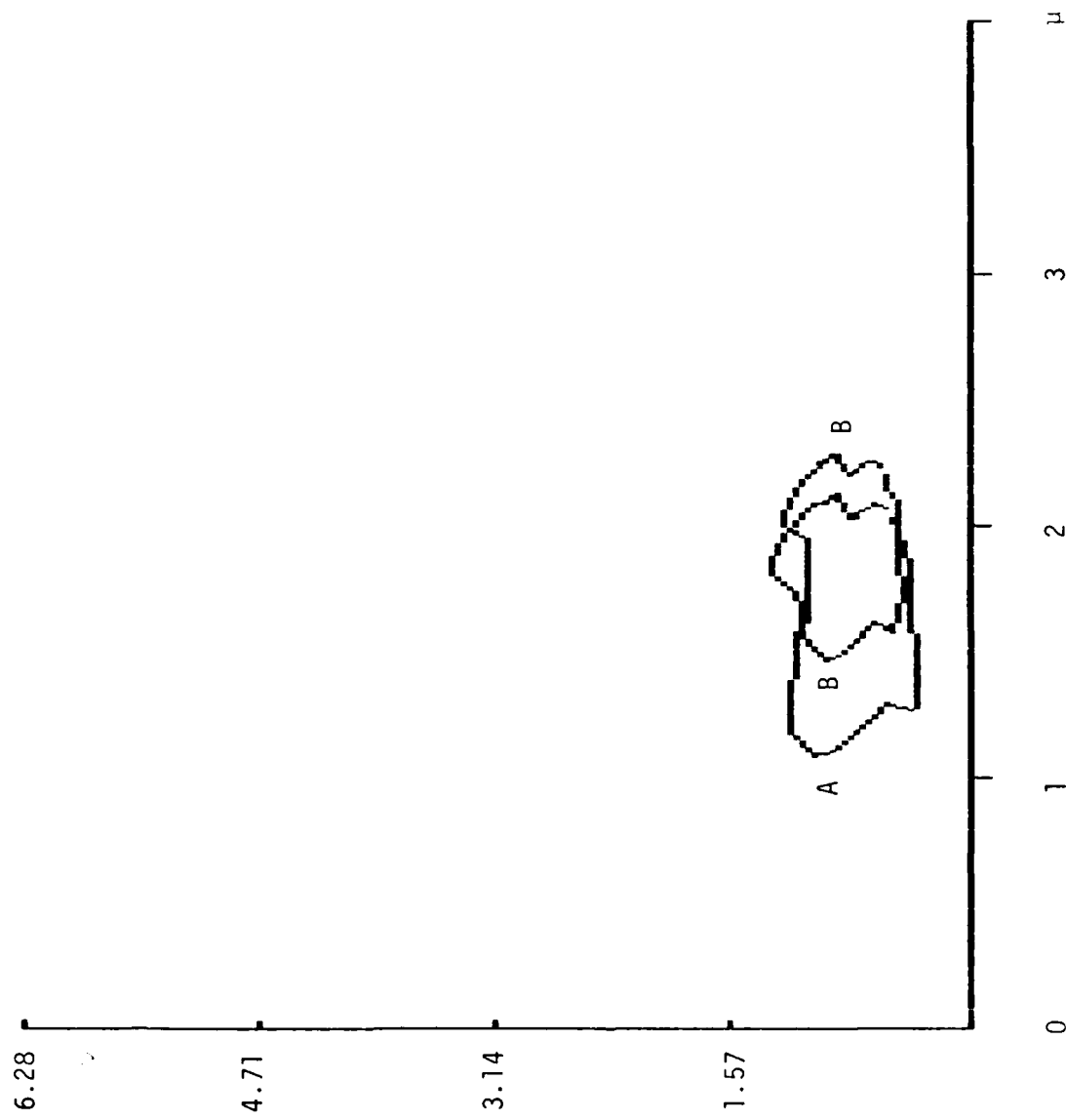


Fig III.23 Off-center random object moving on a line at 45° ; B displaced by (1,1) with respect to A.

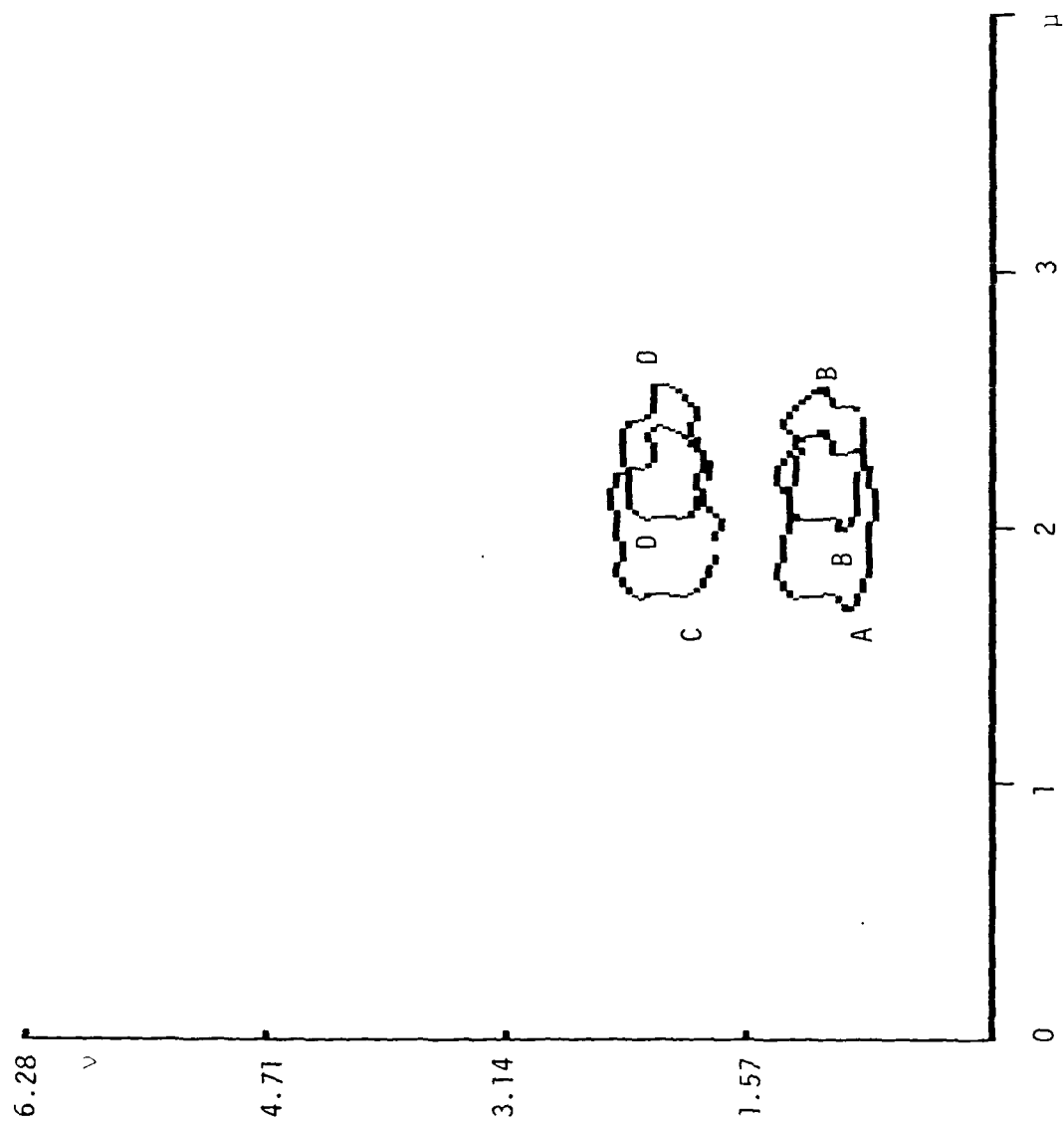


Fig III.24 Off-center random figure moving (a) and a line at 60° with A at (4, 6.93) and B at (5, 8.66); (b) on a line at 120° with C at (-4, 6.93) and D at (-5, 8.66)

The next two steps for rigid motion are:

- a) completing the algorithm to include translation of off-center objects.
- b) developing a program to implement the algorithms. Both of these phases are currently under progress.

Research of dynamic scenes, especially those consisting of non-rigid or elastic motion is being actively pursued in several research centers, and its extension to the new sensor should be seriously considered.

Reference

- [1] Jain, R. "Extraction of Motion Information from Peripheral Process," *IEEE Trans. on PAMI*, Vol. PAMI-3, No. 5, Sept. 1981, pp. 489-503.

CHAPTER IV

ALGORITHMS FOR TARGET TRACKING USING THE BVS SENSOR

IV.1 Introduction

The development of algorithms for a real time, automatic system capable of tracking 2-D targets in complex scenes, is discussed in this chapter. The algorithms developed are based upon the new sensor which, as discussed in previous chapters, produces shape invariance in C.P. under magnification or rotation about the optical axis in I.P. These properties, however, do not apply when an object deviates from the reference location with respect to the origin in the image plane. Significant distortion occurs in the pattern in computation space when objects experience translation in image plane.

In this chapter, three algorithms which can ultimately be used for tracking an object in computation plane are developed. The first algorithm estimates amount of distortion, Δx and Δy , when the object experiences translation only. The second estimates object perturbations such as rotation, dilation, and translation (or any combination of the three). The third algorithm is based on the intensity function and avoids the use of corresponding points.

IV.2 Linear estimation of target perturbations utilizing the logarithmic function

In this section, we develop two techniques which can be used (under some circumstances) to measure small perturbations of target images in computation plane. These techniques are mainly based on the logarithmic function. The first technique applies to a translational case only. The second

applies to the general case of scaling, rotation, and translation.

IV.2.1 Translational case

If two successive image frames are considered, their differences are approximately equal to a linear combination of the components of the displacement of the target. If all points of the frame undergo the same movement, i.e., if motion is rigid, then the perturbation estimation problem is solved using linear estimation. Let Δx and Δy be the components of target displacement in the x and y directions, respectively. By Taylor Series expansion, the frame differences are

$$f(x+\Delta x, y+\Delta y) - f(x, y) = \left. \frac{\partial f(x, y)}{\partial x} \right|_{x=x_n} \Delta x + \left. \frac{\partial f(x, y)}{\partial y} \right|_{y=y_n} \Delta y + \eta(x, y) \quad (1)$$

where

$$\begin{aligned} f(x, y) &= \frac{1}{2} \ln(x^2 + y^2) + j \tan^{-1} \frac{y}{x} \\ f(x+\Delta x, y+\Delta y) &= \frac{1}{2} \ln\{(x+\Delta x)^2 + (y+\Delta y)^2\} + j \tan^{-1} \left(\frac{y+\Delta y}{x+\Delta x} \right) \\ \frac{\partial f(x, y)}{\partial x} &= \frac{1}{x^2 + y^2} (x - jy) \\ \frac{\partial f(x, y)}{\partial y} &= \frac{1}{x^2 + y^2} (y + jx) \end{aligned} \quad (2)$$

The term $\eta(x, y)$ takes into account both various types of noise and the higher order terms of the Taylor Series expansion.

Notice that to write (1) we require $f(x, y)$ to be differentiable. To be differentiable, $f(x, y)$ must be an analytic function in a domain D where

$$D : r > 0 ; 0 < \phi < 2\pi$$

Thus, $f(x, y)$ is always differentiable since its branch cut includes origin and

x -axis where $x > 0$. In this domain, the first order partial derivatives of its component functions must satisfy the Cauchy-Riemann equations [1].

Now letting

$$\alpha = \frac{\partial f(x,y)}{\partial x}$$

$$\beta = \frac{\partial f(x,y)}{\partial y}$$

$$\Delta T = f(x+\Delta x, y+\Delta y) - f(x, y) \quad (3)$$

equation (1) becomes

$$\Delta T(i) = \alpha(i)\Delta x + \beta(i)\Delta y + \eta_i \approx \alpha(i)\Delta x + \beta(i)\Delta y, \quad i = 1, \dots, m \quad (4)$$

where we have included the dependence on x, y in the index i , and m is the number of sampling points.

Since Δx and Δy are constant due to the rigid motion assumption, we can use linear regression to get $\Delta \hat{x}$ and $\Delta \hat{y}$. Introducing an error term (r_i), (4) becomes

$$\Delta T_i = \alpha_i \Delta x + \beta_i \Delta y + r_i \quad (5)$$

or

$$r_i = \Delta T_i - \alpha_i \Delta x - \beta_i \Delta y$$

Use the minimum mean square error method to solve for Δx and Δy .

$$\begin{aligned} |r_i|^2 &= r_i \overline{r_i} \\ &= (\Delta T_i - \alpha_i \Delta x - \beta_i \Delta y)(\Delta \overline{T_i} - \overline{\alpha_i} \Delta x - \overline{\beta_i} \Delta y) \\ &= |\Delta T_i|^2 - \overline{\alpha_i} \Delta x \Delta T_i - \overline{\beta_i} \Delta y \Delta T_i - \beta_i \Delta y \Delta \overline{T_i} - \\ &\quad \alpha_i \Delta x \Delta \overline{T_i} + |\alpha_i|^2 \Delta x^2 + \alpha_i \overline{\beta_i} \Delta x \Delta y + \\ &\quad |\beta_i|^2 \Delta y^2 + \overline{\alpha_i} \beta_i \Delta x \Delta y \end{aligned} \quad (6)$$

For m data points (6) becomes

$$\begin{aligned}
 S &= \sum_{i=1}^m |r_i|^2 \\
 &= \sum_{i=1}^m \left[|\Delta T_i|^2 + |\alpha_i|^2 \Delta x^2 + |\beta_i|^2 \Delta y^2 - 2\operatorname{Re}(\alpha_i \Delta \bar{T}_i) \Delta x - \right. \\
 &\quad \left. 2\operatorname{Re}(\bar{\beta}_i \Delta T_i) \Delta y + 2\operatorname{Re}(\bar{\alpha}_i \beta_i) \Delta x \Delta y \right]
 \end{aligned} \tag{7}$$

Now taking derivatives of S with respect to Δx and Δy , and setting them equal to zero, we obtain

$$\begin{aligned}
 \frac{\partial S}{\partial \Delta x} &= \sum_{i=1}^m \left[2|\alpha_i|^2 \Delta x - 2\operatorname{Re}(\alpha_i \Delta \bar{T}_i) + 2\operatorname{Re}(\bar{\alpha}_i \beta_i) \Delta y \right] \\
 &= 0
 \end{aligned} \tag{8}$$

and

$$\begin{aligned}
 \frac{\partial S}{\partial \Delta y} &= \sum_{i=1}^m \left[2|\beta_i|^2 \Delta y - 2\operatorname{Re}(\bar{\beta}_i \Delta T_i) + 2\operatorname{Re}(\bar{\alpha}_i \beta_i) \Delta x \right] \\
 &= 0
 \end{aligned} \tag{9}$$

Solve (8) and (9) simultaneously for Δx and Δy to obtain

$$\Delta x \approx \Delta \hat{x} = \frac{\sum_{i=1}^m \operatorname{Re}(\alpha_i \Delta \bar{T}_i)}{\sum_{i=1}^m |\alpha_i|^2} \tag{10}$$

$$\Delta y \approx \Delta \hat{y} = \frac{\sum_{i=1}^m \operatorname{Re}(\bar{\beta}_i \Delta T_i)}{\sum_{i=1}^m |\beta_i|^2} \tag{11}$$

where

$$\begin{aligned}
 |\alpha_i|^2 &= |\beta_i|^2 \\
 &= \frac{1}{x_i^2 + y_i^2}
 \end{aligned} \tag{12}$$

and

$$\operatorname{Re}(\bar{\beta}_i \Delta T_i) = -\operatorname{Im}(\alpha_i \Delta \bar{T}_i) \tag{13}$$

Notice that α and β both are expressed in xy coordinates (image plane coordinates). To express them in computational plane coordinates (uv coordinates), we may proceed as follows:

Since

$$x = e^u \cos v \quad (14)$$

$$y = e^u \sin v \quad (15)$$

substitute (14) and (15) into (2). Thus, α and β become

$$\begin{aligned} \alpha &= e^{-(u+jv)} \\ &= e^{-w} \end{aligned} \quad (16)$$

$$\begin{aligned} \beta &= je^{-(u+jv)} \\ &= je^{-w} \end{aligned} \quad (17)$$

and (12) becomes

$$\begin{aligned} |\alpha_i|^2 &= |\beta_i|^2 \\ &= \frac{1}{e^{2u_i}} \end{aligned} \quad (18)$$

Looking at (10) and (11), it is clear that, if the frame difference, ΔT_i , can be measured accurately, the estimation of target displacements should yield accurate results (see computer simulations, sect. IV.3). It should be noted that this particular approach is similar to that proposed in [2].

In the next section we will introduce the algorithm to estimate target displacements in the case of translation, rotation, and scaling.

IV.2.2 Translation, Rotation, and Scaling

This section is concerned with the development of an algorithm for the case of translation, rotation, and scaling. The approach used is similar to that of Schalkoff and McVey [3,4]. They outline the overall problem of

visual target tracking, with emphasis on successive scene target modeling and tracking algorithm. A general 3-D mathematical model for constrained frame-to-frame target evolution is developed. The Taylor Series Video Image Processor (TSVIP) algorithm is suggested.

Following the approach of [3] and [4], it is clear that the frame-to-frame target perturbations can be adapted by using the affine transform described in [5]. This means that target motion is constrained to be observed as 2-D time-varying affine perturbations of x and y in the plane perpendicular to the optical axis. This, however, does not limit target motion to 2-D, since an object moving simultaneously to the right and towards the camera may be modeled by an affine transform representing both dilation and translation. For the sake of simplicity, perturbations such as "roll" of a target whose major axis is in a plane other than the optical axis will not be considered.

Let

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{x}' = A\mathbf{x} + \mathbf{b}$$

where A denotes the homogeneous affine transform matrix and \mathbf{b} the translation vector. Two well-known versions of the homogeneous affine transform are

$$A = \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix}$$

where κ is a scaling factor, and

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

where θ is an angle of rotation about the origin.

$$\mathbf{b} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

The frame difference can then be written as

$$\Delta T(\mathbf{x}, t_2) = \left. \left(\frac{\partial f(\mathbf{x}, t_1)}{\partial \mathbf{x}} \right)^T \right|_{\mathbf{x}=\mathbf{x}_n} \left[\hat{A}(t_2-t_1, t_1) \mathbf{x} + \mathbf{b}(t_2-t_1, t_1) \right] \quad (19)$$

where we define

$$\hat{A}(t_2-t_1, t_1) \triangleq A(t_2-t_1, t_1) - I$$

and

$$\Delta T(\mathbf{x}, t_2) \triangleq f(\mathbf{x}', t_2) - f(\mathbf{x}, t_1)$$

The matrix formulation can be defined as

$$\hat{A} \triangleq \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$\mathbf{b} \triangleq \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$\left. \left(\frac{\partial f(\mathbf{x}, t_1)}{\partial \mathbf{x}} \right)^T \right|_{\mathbf{x}=\mathbf{x}_n} \triangleq [\alpha \quad \beta]$$

$$\mathbf{a} \triangleq \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ \hline \Delta x \\ \Delta y \end{bmatrix}$$

Therefore, (19) can be rewritten as

$$\Delta T(\mathbf{x}_i) = [x_i \alpha_i \quad y_i \alpha_i \quad x_i \beta_i \quad y_i \beta_i \mid \alpha_i \quad \beta_i] \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ \hline \Delta x \\ \Delta y \end{bmatrix} \quad (20)$$

From (20) it is apparent that we can solve for \mathbf{a} by using a least square estimation approach as we did in the previous section. The most obvious objection is that it requires the inversion of a 6×6 matrix (or equivalently the solution of the six corresponding normal equations). As mentioned earlier in this section, the target and camera perturbations are restricted to rotation, dilation, and translation (or any combination of the three) in the plane perpendicular to the optical axis. In this case we can let

$$a_1 = a_{11} = a_{22}$$

and

$$a_2 = a_{12} = -a_{21}$$

and (20) can then be rewritten as

$$\Delta T(\mathbf{x}_i) = [x_i \alpha_i + y_i \beta_i \quad x_i \beta_i - y_i \alpha_i \mid \alpha_i \quad \beta_i] \begin{bmatrix} a_1 \\ a_2 \\ \Delta x \\ \Delta y \end{bmatrix} \quad (21)$$

Written more compactly, (21) yields

$$\Delta T(\mathbf{x}_i) = [H \mid G] \mathbf{a} \quad (22)$$

where

$$G = [\alpha_i \quad \beta_i]$$

$$H = [x_i \alpha_i + y_i \beta_i \quad x_i \beta_i - y_i \alpha_i]$$

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \mathbf{a}_c \\ \mathbf{b} \end{bmatrix}$$

or,

$$\Delta T = H \mathbf{a}_c + G \mathbf{b} \quad (23)$$

Thus, the problem is now: given (22), generate a "good" estimate of \mathbf{a} . Again, the least squares solution appears promising. Let's introduce an error term (\mathbf{r}_i). Then (23) becomes

$$\Delta \mathbf{T}_i = H_i \mathbf{a}_c + G_i \mathbf{b} + \mathbf{r}_i$$

$$i = 1, \dots, m \quad (24)$$

or

$$\mathbf{r}_i = \Delta \mathbf{T}_i - H_i \mathbf{a}_c - G_i \mathbf{b} \quad (25)$$

Notice that H and G are complex matrices and their dimensions are $N \times 2$. $\Delta \mathbf{T}$ and \mathbf{r} are also complex matrices and both of them have dimensions of $N \times 1$. \mathbf{a}_c and \mathbf{b} , however, are real matrices and their dimensions are 2×1 . Dropping subscript i and substituting H , G , \mathbf{a}_c , and \mathbf{b} into (25), we obtain

$$r = \Delta T - (x\alpha + y\beta)a_1 - (x\beta - y\alpha)a_2 - \alpha\Delta x - \beta\Delta y \quad (26)$$

The conjugate of r can be written easily as

$$\bar{r} = \Delta \bar{T} - (x\bar{\alpha} + y\bar{\beta})a_1 - (x\bar{\beta} - y\bar{\alpha})a_2 - \bar{\alpha}\Delta x - \bar{\beta}\Delta y \quad (27)$$

Now we can use the minimum mean square error method to solve for \mathbf{a}_c and \mathbf{b} .

$$r\bar{r} = \{ \Delta T - (x\alpha + y\beta)a_1 - (x\beta - y\alpha)a_2 - \alpha\Delta x - \beta\Delta y \}$$

$$\{ \Delta \bar{T} - (x\bar{\alpha} + y\bar{\beta})a_1 - (x\bar{\beta} - y\bar{\alpha})a_2 - \bar{\alpha}\Delta x - \bar{\beta}\Delta y \} \quad (28)$$

Expand and simplify (28) for m data points. Thus, we obtain

$$S = \sum_{i=1}^m r_i \bar{r}_i$$

$$\begin{aligned}
 = & \sum_{i=1}^m \left[|\Delta T_i|^2 - 2\operatorname{Re}(x_i \alpha_i + y_i \beta_i) a_1 \Delta \bar{T}_i - 2\operatorname{Re}(x_i \beta_i - y_i \alpha_i) a_2 \Delta \bar{T}_i - 2\operatorname{Re}(\alpha_i \Delta x \Delta \bar{T}_i) - \right. \\
 & 2\operatorname{Re}(\bar{\beta}_i \Delta y \Delta T_i) + 2\operatorname{Re}[\alpha_i \Delta x (x_i \bar{\alpha}_i + y_i \bar{\beta}_i)] a_1 + a_1^2 + \\
 & 2\operatorname{Re}[\beta_i \Delta y (x_i \bar{\alpha}_i + y_i \bar{\beta}_i)] a_1 + 2\operatorname{Re}[\alpha_i \Delta x (x_i \bar{\beta}_i - y_i \bar{\alpha}_i)] a_2 + a_2^2 + \\
 & \left. 2\operatorname{Re}[\beta_i \Delta y (x_i \bar{\beta}_i - y_i \bar{\alpha}_i)] a_2 + 2\operatorname{Re}(\bar{\alpha}_i \beta_i) \Delta x \Delta y + \alpha_i \bar{\alpha}_i \Delta x^2 + \beta_i \bar{\beta}_i \Delta y^2 \right]
 \end{aligned} \tag{29}$$

Now taking derivatives of S with respect to a_1 , a_2 , Δx , and Δy and setting them equal to zero, we obtain 4 equations for 4 unknowns:

$$\left(\sum_{i=1}^m \frac{x_i}{x_i^2 + y_i^2} \right) a_1 - \left(\sum_{i=1}^m \frac{y_i}{x_i^2 + y_i^2} \right) a_2 + \left(\sum_{i=1}^m \frac{1}{x_i^2 + y_i^2} \right) \Delta x = \operatorname{Re} \sum_{i=1}^m \alpha_i \Delta \bar{T}_i \tag{30}$$

$$\left(\sum_{i=1}^m \frac{y_i}{x_i^2 + y_i^2} \right) a_1 + \left(\sum_{i=1}^m \frac{x_i}{x_i^2 + y_i^2} \right) a_2 + \left(\sum_{i=1}^m \frac{1}{x_i^2 + y_i^2} \right) \Delta y = \operatorname{Re} \sum_{i=1}^m \bar{\beta}_i \Delta T_i \tag{31}$$

$$a_1 + \left(\sum_{i=1}^m \frac{x_i}{x_i^2 + y_i^2} \right) \Delta x + \left(\sum_{i=1}^m \frac{y_i}{x_i^2 + y_i^2} \right) \Delta y = \operatorname{Re} \sum_{i=1}^m \Delta \bar{T}_i \tag{32}$$

$$a_2 - \left(\sum_{i=1}^m \frac{y_i}{x_i^2 + y_i^2} \right) \Delta x + \left(\sum_{i=1}^m \frac{x_i}{x_i^2 + y_i^2} \right) \Delta y = \operatorname{Im} \sum_{i=1}^m \Delta \bar{T}_i \tag{33}$$

Solve (30), (31), (32), and (33) simultaneously for a_1 , a_2 , Δx , and Δy . Thus, we obtain

$$a_1 \approx \hat{a}_1 = \frac{-q_1 P + q_3 S + q_4 R}{\text{diff}} + 1 \tag{34}$$

$$a_2 \approx \hat{a}_2 = \frac{-q_3 R + q_4 S - q_2 P}{\text{diff}} \tag{35}$$

$$\Delta x \approx \Delta \hat{x} = \frac{q_1 S - q_3 - q_2 R}{\text{diff}} \tag{36}$$

$$\Delta y \approx \Delta \hat{y} = \frac{q_1 R - q_4 + q_2 S}{\text{diff}} \tag{37}$$

where

$$\begin{aligned}
 S &= \sum_{i=1}^m \frac{x_i}{x_i^2 + y_i^2} \\
 R &= \sum_{i=1}^m \frac{y_i}{x_i^2 + y_i^2} \\
 P &= \sum_{i=1}^m \frac{1}{x_i^2 + y_i^2} \\
 q_1 &= \operatorname{Re} \sum_{i=1}^m \Delta \bar{T}_i \\
 q_2 &= \operatorname{Im} \sum_{i=1}^m \Delta \bar{T}_i \\
 q_3 &= \operatorname{Re} \sum_{i=1}^m \alpha_i \Delta \bar{T}_i \\
 q_4 &= \operatorname{Re} \sum_{i=1}^m \bar{\beta}_i \Delta T_i \\
 diff &= S^2 + R^2 - P
 \end{aligned} \tag{38}$$

Again, we may express (38) in uv coordinates as

$$\begin{aligned}
 S &= \sum_{i=1}^m e^{-u_i} \cos v_i \\
 R &= \sum_{i=1}^m e^{-u_i} \sin v_i \\
 P &= \sum_{i=1}^m \frac{1}{e^{2u_i}}
 \end{aligned} \tag{39}$$

Notice that if a target experiences only translation, then (36) and (37) are identical to (10) and (11). The scaling factor, κ , can be estimated as

$$\kappa = \sqrt{\hat{a}_1^2 + \hat{a}_2^2} \tag{40}$$

Thus, the above results may be used to track the reference point on the target if we can measure the frame difference accurately. This is the so called "correspondence problem" which is under consideration by a number of researchers [9,10,11] and for which a general solution does not exist yet. For the interest of verifying the theoretical results, we will assume that the frame difference can be measured exactly. In other words, the corresponding

point problem is assumed to be solved.

IV.3 Computer Simulation

Computer simulation was performed to lend credence to the theoretical results, in particular the validity of the algorithm in section IV.2.2. A rectangle of 4×6 arbitrary units of length was used to represent the target, thus this rectangle may be perturbed exactly via the affine transform. A 32×32 (number of sampling points, m , equals to 32) scene was used. The results are as follows:

Figs. IV.1(a) and IV.1(b) compare exact and algorithm estimates of target translational parameters, i.e., Δx and $\Delta \hat{x}$, and Δy and $\Delta \hat{y}$ for the case of translation only. In this case, Δx was varied in a range of -2.8 to 2.91 and Δy was varied in a range of -1.8 to 1.82 . As shown, the algorithm translational estimates are quite good over the specified range of Δx and Δy above. The corresponding errors, i.e., $\Delta x - \Delta \hat{x}$ and $\Delta y - \Delta \hat{y}$ are shown in Figs. IV.2(a) and IV.2(b).

The algorithm ((36) and (37) were used) estimates target translational parameters in the case of rotation, translation, and dilation. The target first rotates about the origin and then translates. The angle of rotation, θ , was varied from -15 degrees to $+15$ degrees. The dilation parameter, κ , was set equal to unity. The exact translational parameters, Δx and Δy , both were set the same as before. Figs. IV.3(a) and IV.3(b) show exact and algorithm estimates of target translational parameters versus the angle of rotation. The results are quite good over a range of -5 degrees to $+5$ degrees (see also

Figs. IV.4(a) and IV.4(b) for error versus the angle). The algorithm tends to break down when the perturbations get larger. This agrees with theory since we only keep the first order term of Taylor Series. Thus, the algorithm is expected to work well for small perturbations.

Figs. IV.5(a) and IV.5(b) show the error versus number of points, i.e., number of sampling points. The results verify that as the number of sampling points increases, the error should tend to decrease. However, from the graphs shown, we see that the error at a smaller number of sampling points is less than the one at a bigger number of sampling points. This is due to the geometry of the target itself. For the overall graph, however, the error tends to decrease as expected.

The algorithm presented in this section appears to be implementable and temporally efficient for real-time video tracking using the new sensor. By capitalizing on successive frame difference temporal dependence, the algorithm estimates target rotation, dilation, and translation parameters. Computer simulation using the 4×6 rectangle to represent the target confirms the validity of the algorithm. Various issues such as matrix inversion problem, error analysis, corresponding point problem, etc., have not yet been solved. The matrix inversion problem and error analysis can be solved easily. For example, we can use a generalized inverse matrix approach to solve the inversion problem [6]. However, the correspondence problem, still under investigation, is much more difficult and other techniques for target tracking will be considered.

IV.4 A tracking algorithm using the intensity function

Since the corresponding point problem appears to be very difficult to solve, a new approach is used to derive a tracking algorithm. This approach utilizes the intensity function and does not require to have the solution of the corresponding point problem. The technique used here is similar to that of Netravali and Robbins [7]. The other technique which also avoids correspondence problem is proposed by Stuller and Paik [8]. Their approach is based on the Kalman Filter.

For the sake of simplicity, we will consider the translation case only. Based on the intensity function, we will derive a simple recursive algorithm to estimate the displacement vector (\hat{D}). The algorithm seeks to minimize a function of the prediction error by iterating in a gradient or steepest descent direction such that the consecutive estimates converge to an estimate of the translation vector.

Let $J(\mathbf{x}, t)$ and $J(\mathbf{x}, t - \tau)$ be the intensity values of the two successive frames as a function of spatial location \mathbf{x} (a two-dimensional vector) and time t . The time between the two frames is τ . We now define the quantity $\Delta J(\mathbf{x}, \hat{D}^{i-1})$ as

$$\Delta J(\mathbf{x}, \hat{D}^{i-1}) = J(\mathbf{x}, t) - J(\mathbf{x} - \hat{D}^{i-1}, t - \tau) \quad (41)$$

Since

$$\begin{aligned} J(\mathbf{x}, t) &= J(\mathbf{x} - D, t - \tau) \\ J(\mathbf{x} + D, t) &= J(\mathbf{x}, t - \tau) \\ J(\mathbf{x} + D - \hat{D}^{i-1}, t) &= J(\mathbf{x} - \hat{D}^{i-1}, t - \tau) \end{aligned} \quad (42)$$

We can rewrite (41) as

$$\begin{aligned}\Delta J(\mathbf{x}, \hat{D}^{i-1}) &= J(\mathbf{x}, t) - J(\mathbf{x} + D - \hat{D}^{i-1}, t) \\ &= -(D - \hat{D}^{i-1})^T \nabla J(\mathbf{x}, t) + \text{hot}\end{aligned}\quad (43)$$

Now let $D - \hat{D}^{i-1}$ be an error at the $(i-1)$ th frame (E^i). That is

$$E^i = D - \hat{D}^{i-1} \quad (44)$$

where

$$E^i = (e_x^i \ e_y^i)^T \quad (45)$$

For the i th frame, displacement estimate \hat{D}^i may be obtained by linearizing the intensity function around the displacement estimate for the $(i-1)$ th field.

Let's write \hat{D}^i as

$$\hat{D}^i = \hat{D}^{i-1} + E^i \quad (46)$$

where

$$\hat{D}^i = (\Delta \hat{x}^i \ \Delta \hat{y}^i)^T \quad (47)$$

Rewrite (43) as

$$\Delta J(\mathbf{x}, \hat{D}^{i-1}) \approx -E^{iT} \nabla J(\mathbf{x}, t) \quad (48)$$

The gradient J ($\nabla J(\mathbf{x}, t)$) can be evaluated as follows:

$$\begin{aligned}\nabla J(\mathbf{x}, t) &= \nabla I(w, t) \\ &= \nabla I(\log z, t) \\ &= \nabla I(\log(x + jy), t) \\ &= \frac{\partial I(\log(x + jy), t)}{\partial x} + \frac{\partial I(\log(x + jy), t)}{\partial y}\end{aligned}\quad (49)$$

By the chain rule, we can write $\frac{\partial I}{\partial x}$ as

$$\frac{\partial I}{\partial x} = \frac{\partial I}{\partial u} \frac{\partial u}{\partial x} + \frac{\partial I}{\partial v} \frac{\partial v}{\partial x} \quad (50)$$

and

$$\begin{aligned}\frac{\partial I}{\partial u} &\approx \frac{1}{2} \left[\frac{I(w+\Delta u) - I(w)}{\Delta u} + \frac{I(w) - I(w-\Delta u)}{\Delta u} \right] \\ \frac{\partial I}{\partial v} &\approx \frac{1}{2} \left[\frac{I(w+\Delta v) - I(w)}{\Delta v} + \frac{I(w) - I(w-\Delta v)}{\Delta v} \right]\end{aligned}\quad (51)$$

Similarly,

$$\frac{\partial I}{\partial y} = \frac{\partial I}{\partial u} \frac{\partial u}{\partial y} + \frac{\partial I}{\partial v} \frac{\partial v}{\partial y} \quad (52)$$

and the remaining partial derivatives can be evaluated as

$$\begin{aligned}\frac{\partial u}{\partial x} &= \frac{x}{x^2+y^2} \\ \frac{\partial v}{\partial x} &= \frac{-y}{x^2+y^2} \\ \frac{\partial u}{\partial y} &= \frac{y}{x^2+y^2} \\ \frac{\partial v}{\partial y} &= \frac{x}{x^2+y^2}\end{aligned}\quad (53)$$

Now we can solve (48) by using the least square estimation technique. Let's rewrite (48) as

$$\Delta J(\mathbf{x}, \hat{D}^{i-1}) \approx -(e^i_x \frac{\partial I}{\partial x} + e^i_y \frac{\partial I}{\partial y}) \quad (54)$$

Solve (54) for e^i_x and e^i_y by using the same procedure as shown in section IV.2.1. The solutions are

$$\begin{aligned}e^i_x &= -\frac{ac - fd}{diff} \\ e^i_y &= -\frac{bd - fc}{diff}\end{aligned}\quad (55)$$

where a,b,c,d,f and *diff* are defined below:

$$\begin{aligned}
 a &= \sum_{k=1}^m \beta_k^2 \\
 b &= \sum_{k=1}^m \alpha_k^2 \\
 c &= \sum_{k=1}^m \alpha_k \Delta J_k \\
 d &= \sum_{k=1}^m \beta_k \Delta J_k \\
 f &= \sum_{k=1}^m \alpha_k \beta_k \\
 diff &= \sum_{k=1}^m \alpha_k^2 \sum_{k=1}^m \beta_k^2 - \left(\sum_{k=1}^m \alpha_k \beta_k \right)^2
 \end{aligned} \tag{56}$$

where

$$\begin{aligned}
 \alpha &= \frac{\partial I}{\partial x} \\
 \beta &= \frac{\partial I}{\partial y} \\
 k &= 1, \dots, m
 \end{aligned} \tag{57}$$

where m is the number of sampling point. Combine (46) and (55) and obtain

$$\begin{aligned}
 \Delta \hat{x}^i &= \Delta \hat{x}^{i-1} + e_x^i \\
 \Delta \hat{y}^i &= \Delta \hat{y}^{i-1} + e_y^i
 \end{aligned} \tag{58}$$

Notice that $\Delta J(\mathbf{x}, \hat{D}^{i-1})$ is defined in terms of two quantities: (i) the spatial location \mathbf{x} at which it is evaluated and (ii) the displacement \hat{D}^{i-1} with which it is evaluated. Obviously, we do not need to know what point corresponds to what point in order to evaluate $\Delta J(\mathbf{x}, \hat{D}^{i-1})$. However, we have to choose an initial estimate of D (i.e.; \hat{D}^0) to start the iterative process. The consecutive estimates should converge to an estimate of the translation vector. At this point in time, neither a proof of convergence nor the performance of the algorithm above have yet been demonstrated. Both subjects are currently in progress.

IV.5 Conclusion and Plans for the future

In this chapter we have developed algorithms for target tracking using the new sensor for the cases of translation, rotation and scaling using two different techniques:

- a) Affine transformations and the least square error.
- b) Intensity functions (of space and time) and the Kalman filter.

The first technique is mathematically established, for small target displacements, but requires the solution of the correspondence problem, i.e. a knowledge of corresponding points in consecutive images. The second technique avoids the correspondence problem, but the convergence of the algorithm in general cases has not been established.

Simple computer simulations using rectangular targets have shown that the algorithm performs satisfactorily under ideal conditions.

The correspondence problem for technique (a) and the algorithm convergence for technique (b) are nontrivial research areas requiring further significant work.

Another area of current interest which needs to be addressed in our work is non rigid motion which comprises single targets rotating on axis other than the optical axis and multiple targets moving at different speeds.

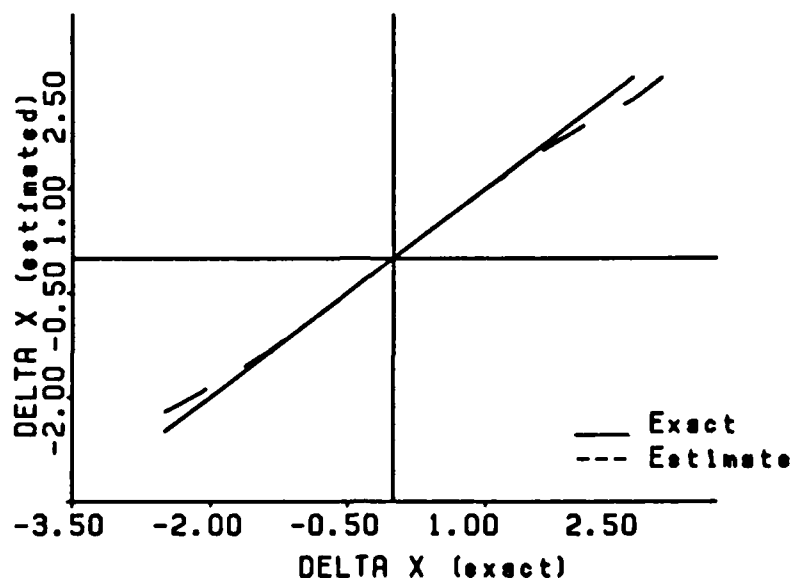


Figure IV.1(a). A horizontal perturbation ($\Delta\hat{x}$) of a rectangle 4 by 6.(Translation case)

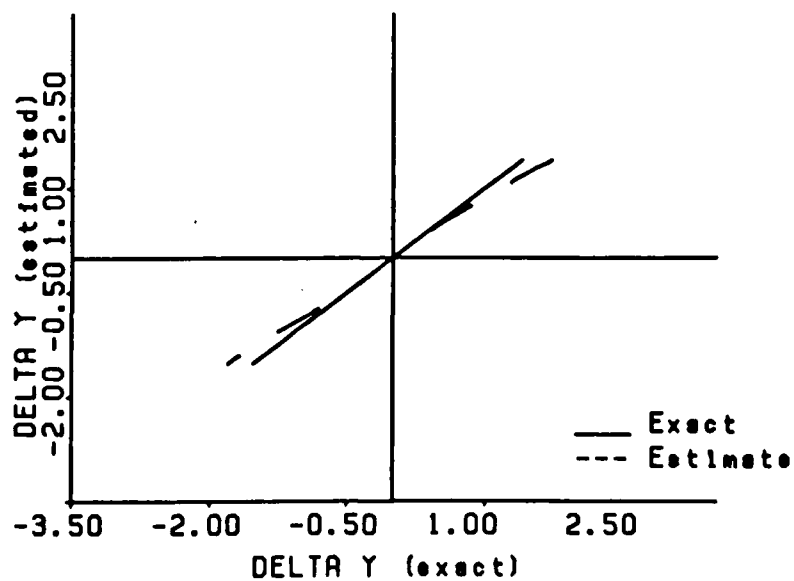


Figure IV.1(b). A vertical perturbation ($\Delta\hat{y}$) of a rectangle 4 by 6 (Translation case)

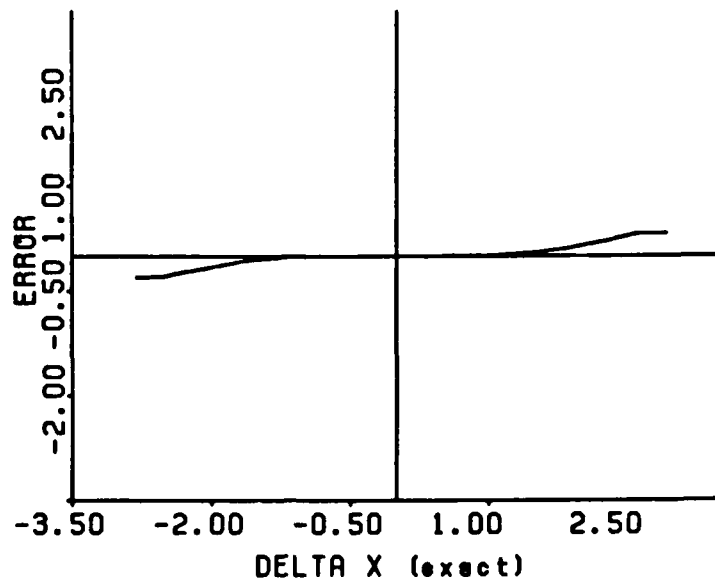


Figure IV.2(a). A horizontal error ($\Delta x - \Delta \hat{x}$) for Fig. IV.1(a)

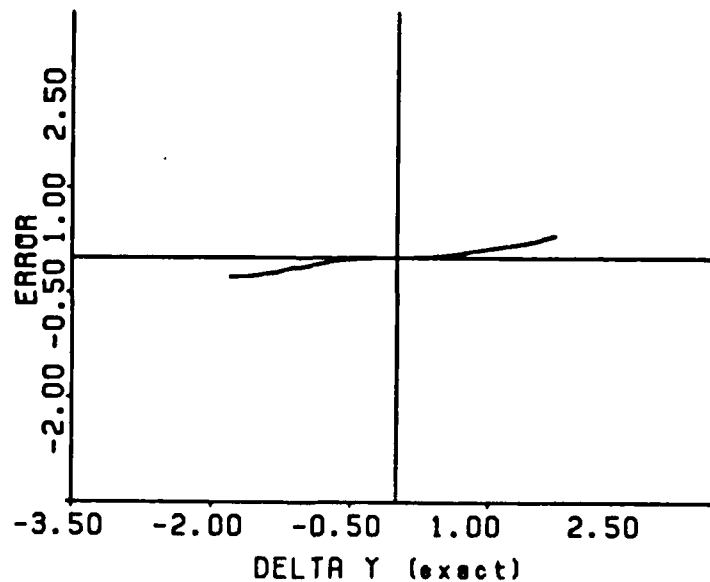


Figure IV.2(b). A vertical error ($\Delta y - \Delta \hat{y}$) for Fig. IV.1(b)

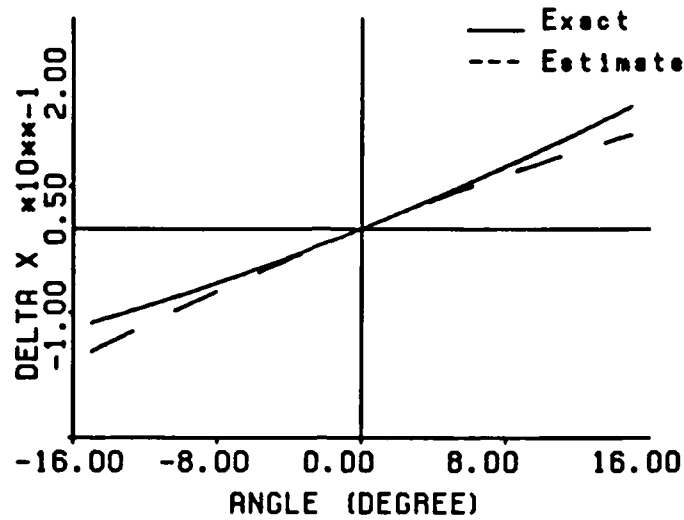


Figure IV.3(a). A horizontal perturbation ($\Delta \hat{x}$) of a rectangle 4 by 6 (Translation, Rotation and Scaling case)

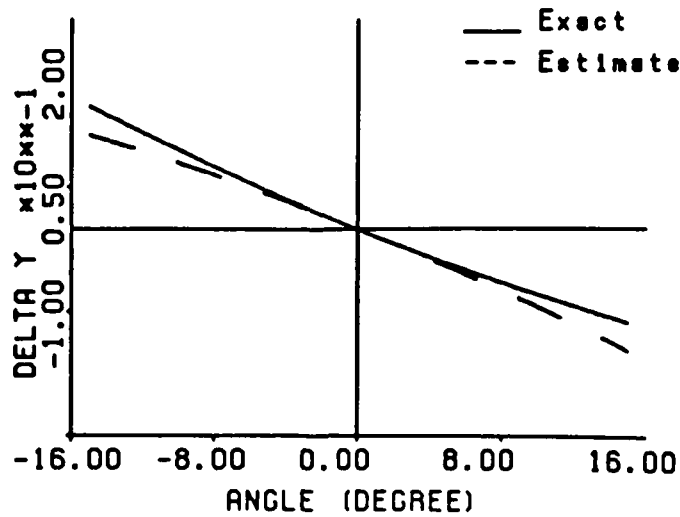


Figure IV.3(b). A vertical perturbation ($\Delta \hat{y}$) of a rectangle 4 by 6 (Translation, Rotation and Scaling case)

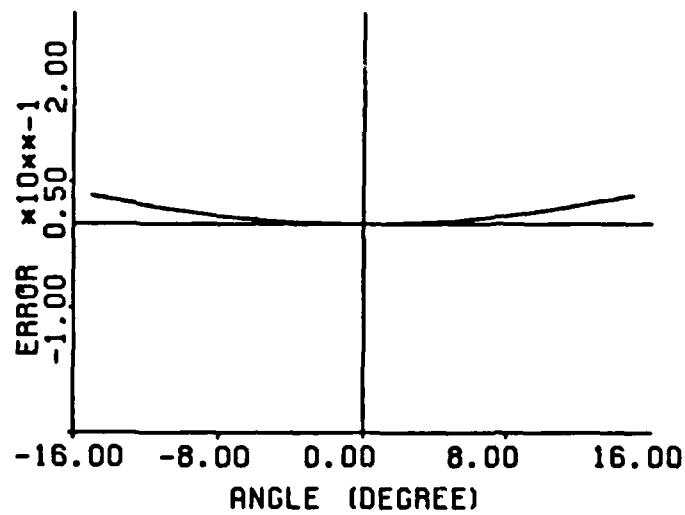


Figure IV.4(a). A horizontal error ($\Delta x - \Delta \hat{x}$) for Fig. IV.3(a)

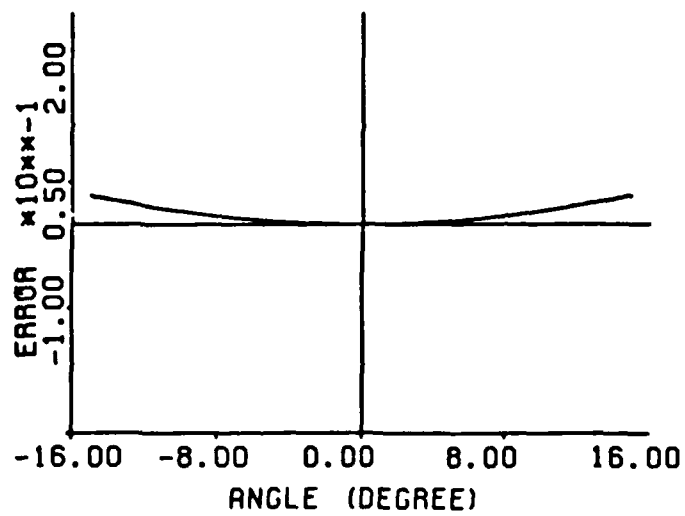


Figure IV.4(b). A vertical error ($\Delta y - \Delta \hat{y}$) for Fig. IV.3(b)

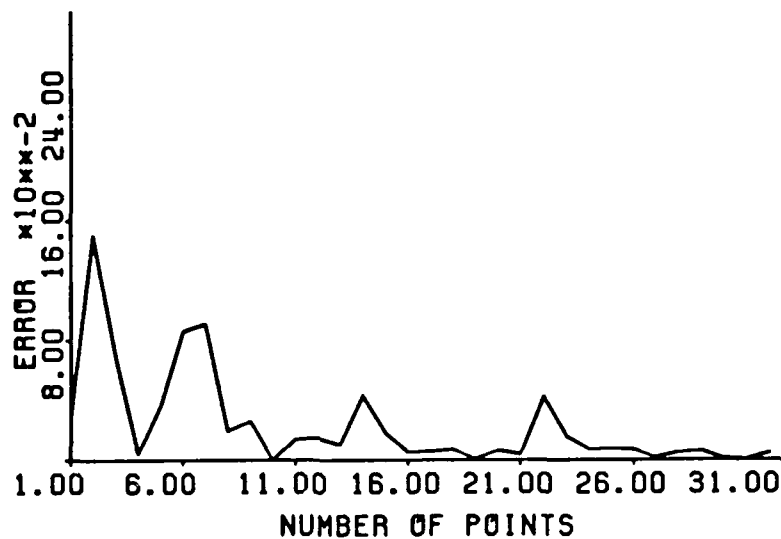


Figure IV.5(a). A horizontal error ($\Delta x - \Delta \hat{x}$) versus number of sampling points .

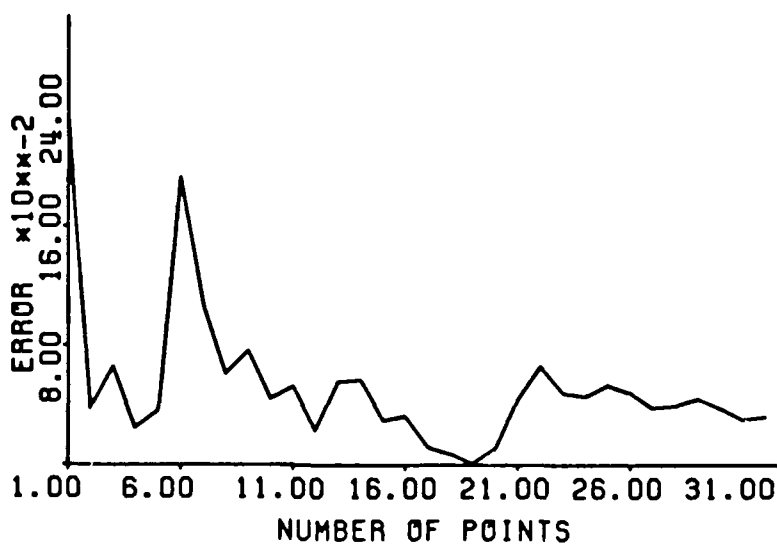


Figure IV.5(b). A vertical error ($\Delta y - \Delta \hat{y}$) versus number of sampling points

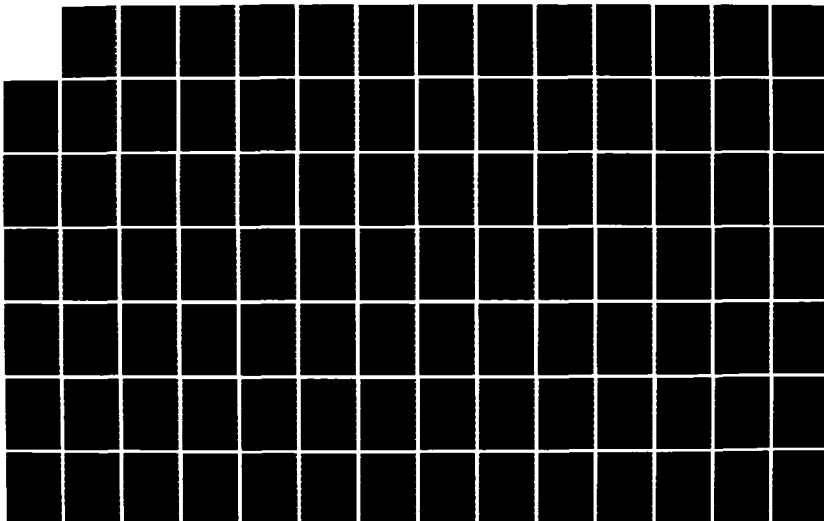
AD-A168 521

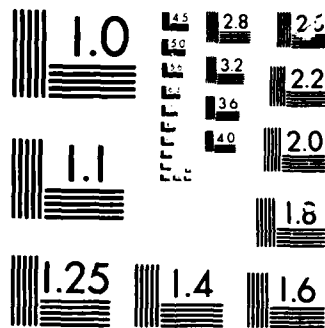
BIOLOGICAL VISUAL SYSTEMS STRUCTURES FOR MACHINE VISION 3/4
APPLIED TO ROBOTI.. (U) VIRGINIA UNIV CHARLOTTESVILLE
DEPT OF ELECTRICAL ENGINEERING.. R M INIGO ET AL.

UNCLASSIFIED

FEB 86 UVA/525647/EE86/101 AFOSR-TR-86-0282 F/G 6/4

NL





BIBLIOGRAPHY

- [1] R. V. Churchill and J. W. Brown, Complex Variables and Application. Fourth Edition. McGraw-Hill Book Co. New York, 1984.
- [2] C. Cafforio and F. Rocca, "Methods for Measuring Small Displacements of Television," IEEE Trans. Inform. Theory, vol. IT-22, pp.573-579, Sept. 1976.
- [3] R. J. Schalkoff and E .S. McVey, "A Model and Tracking Algorithm for a Class of Video targets," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-4, No.1, pp.500-507, Jan. 1982.
- [4] R. J. Schalkoff, "Algorithms for a real-time automatic video tracking system," Ph.D dissertation, Univ. Virginia, Charlottesville, May 1979.
- [5] D. Gans, Transformations and Geometries. New York: Appleton-Century-Crofts, 1969, ch.IV.
- [6] J. Jones, Jr., "Handout on Generalized Inverse for Numerical Analysis, MA 508," Department of Mathematic. Air Force Institute of Technology, Wright-Patterson AFB., Ohio. December, 1984.
- [7] A.N. Netravali and J.D. Robbins, "Motion-Compensated Television Coding: Part I," Bell Syst. Tech. J. (USA). vol.58, No.3 pp.631-70, 1979.
- [8] J.A. Stuller and Hyuang Whan Paik, "Pel-Recursive Television Image Motion Estimate Using The Kalman Filter," Proceedings of Medcomp '82. First IEEE Computer Society International Conference on Medical Computer Science/Computational Medicine. Philadelphia, Pa., USA. New York, USA. XV&547. 181-5. 13. IEEE. 23-25 Sept. 1982.
- [9] Jain Ramesh, "Extraction of Motion Information from Peripheral Processes," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 5, pp.489-503, September 1981.
- [10] Kass, Michael, "Computing Visual Correspondence," Image Understanding: Proceedings of a Workshop (14th) Held at Arlington, Virginia on June 23, 1983, AD-A130 251.
- [11] W.E.L. Grimson, "Computing Surface Shape Using a Theory of Human Stereo Vision," Proceedings of COMPSAC 81. IEEE Computer Society's Fifth International Computer Software and Applications Conference. pp.209-10 16-20 Nov. 1981 Chicago, IL, USA.

CHAPTER V

THE ROLE OF MICROSACCADIC EYE MOVEMENT IN THE HUMAN VISUAL SYSTEM

The purpose of this research is to study microsaccadic eye movement and its role, beneficial or otherwise, in the human visual system. Although the characteristics of microsaccades are well documented, their purpose remains a topic of research, and it is not within the scope of this project to answer the question of their purpose conclusively. Rather, in this report, a hypothesis is drawn describing a possible purpose of microsaccades, and support of the hypothesis is offered in the form of documentation from previous research. The hypothesis is that microsaccades serve to sweep visual stimuli across several receptors (cones) in the fovea. This frequent movement as the eye attempts to focus on its target may help to compensate for inoperative or less sensitive receptors. If the hypothesis proves sound, then the benefits of emulating this type of motion, in terms of improving fault-tolerance and local calibration in electronic detection systems, is clear. Because the documentation presented in this report provides a reasonably sound basis for continuation of the project, the next step, modelling of this portion of the visual system for further study, and eventual possible implementation in hardware, will be taken.

Saccades are rapid, jerklike movements which the eye uses to locate and focus on a target. Although the boundary between what is termed saccadic and microsaccadic movements is unclear, in general, movements that subtend an arc of 10 minutes or less across the eye's surface are termed microsaccadic. These movements, occur naturally at a frequency of 1-2 per second,

and they occur without the perception of image destabilization [1]

It has been implied that microsaccadic movements are involuntary, since they occur during periods of attempted fixation. However, Steinman [2] found through subsequent research and experimentation that microsaccades were largely controllable. He found that when subjects were instructed to fixate on a target, moving their eyes as little as possible, they were able to suppress microsaccades by approximately a factor of 4 [2]. In further research performed by Haddad and Steinman [3] it was found that in the presence of visible target, subjects could voluntarily make microsaccades in the desired direction, and that when spontaneous movements did occur, that subjects were aware of them. These findings were significant because they tended to contradict the idea that microsaccades might be myocardial "noise" [4], or that they occurred strictly in response to drift, and to support the idea that they were part of a more complex function of the human visual system.

A more pronounced relationship between saccadic and microsaccadic eye movement is evident in the work of Zuber and Stark [5,6]. Zuber studied the velocity-amplitude relationship for saccadic movements ranging in size from 1 to 35 degrees, and for microsaccadic movements ranging from 1 to 30 minutes of arc. Two observations were made in these experiments. The first is that, with saturation at higher amplitudes, the maximum velocity of microsaccades increased as function of amplitude of movement. The other, perhaps more significant observation, in terms of supporting the original hypothesis, is that when the data describing the velocity-amplitude relationship of microsaccadic movement was extrapolated directly onto a similar

curve describing the behavior of saccadic movement, the curve remained continuous [5]. This finding supported the idea that saccadic and microsaccadic movement, both voluntary and spontaneous were produced by a common physiological system.

Further support of the common origin of saccadic and microsaccadic movement was found in the results of the Van Gisbergen [7]. He made observations concerning the velocity-amplitude relationship for saccadic and microsaccadic movement that paralleled those made by Zuber and Stark. Van Gisbergen went beyond this initial observation and, through his research and experimentation with rhesus monkeys, provided evidence that a control system, involving burst neurons in the pontomedullary reticular formation, and motoneurons in the abducens nucleus, existed with the visual system to produce saccadic and microsaccadic movement. He trained the monkeys to follow jumping visual targets: the monkeys produced saccades in the range 2-20 degrees and microsaccades in the range 12-30 minutes. He measured action potentials from both burst neurons and motoneurons, during each eye movement by the monkeys. He found evidence that both saccades and microsaccades were generated by burst neurons. He also found evidence to suggest that the burst neuron discharge rate was directly related to motor error, i.e., the difference between the eye's position at any instant, and the eye's desired position, determined from within a higher visual center. He developed a model for a visual control system, whereby burst neurons generated saccades in response to motor error by driving motoneurons in push-pull by ipsilateral and contralateral burst cells.

This research provides more evidence that saccades and microsaccades have a common origin, which again tends to negate the theory that microsaccades are an unwanted byproduct of the visual system, or that they serve no useful purpose [1,4]. However, these observations also tend to imply that both saccadic and microsaccadic movement share a common function, which has been generally shown not to be the case. The function of saccadic movements is well defined. They are required for target detection, retinal slip compensation, and repolarization of sensory neurons in the fovea [8]. However, movements of a velocity lower than that in naturally occurring microsaccades are necessary to perform these functions. It is therefore assumed that, despite evidence of a common physiological origin, and the probability of overlapping functions, that microsaccades serve visual functions that are unique.

Research and experimentation into the relationship between microsaccades and binocular rivalry may have yielded one unique function of microsaccades. Howard Sabrin [9] investigated the effect of visual stimuli which cause binocular rivalry, on the incidence of microsaccadic movement. Binocular rivalry occurs when an observer is presented with stimuli that cannot be fused into one stable image. The result is the alternating dominance of one image over the other. Sabrin found that microsaccades increased by approximately 50% when measured during binocular rivalry. The level of microsaccadic activity was found to increase at the beginning of the suppression interval, and then, as the eye under observation again became dominant, to decrease to a level approximately equal to that seen during normal viewing conditions. Further experimentation showed that the suppressed eye, not the

dominant eye, was responsible for the increase in microsaccadic activity [10].

Because visual thresholds for the suppressed eye increase by .5 log unit [11], the visual control system may interpret the suppression interval as a loss of the visual target, and respond with increased microsaccadic movement. As microsaccades provide information on a neural level, the motor error signal to the oculomotor system, and the resultant microsaccadic movements could decrease [9].

Sabrin [9] continued his experimentation in this area in 1983. He simulated microsaccadic movement in periods of suppression during binocular rivalry to determine whether or not microsaccades actually facilitated dominance of the suppressed stimulus. He found that the simulated movements (8.25 minutes, 0-6/sec) caused dominance in the suppressed eye to increase. Also significant was his finding that the shift from suppression to dominance was facilitated optimally when the emulated microsaccades were of the same frequency as those which occur naturally (1-2/sec).

The results of Sabrin's experimentation are in accordance with Van Gisbergen's model of the oculomotor control system. Sabrin's results also lend support to the original hypothesis, since they imply that microsaccades facilitate the passing of visual information to the brain, despite the increased visual threshold (lower sensitivity of receptors) in the suppressed eye.

The work of Armington [12] provides strong support for the project hypothesis. He investigated the relationship between the amplitude of microsaccadic movement and visual response, by comparing microsaccade lengths (generally <10 minutes) to the visually evoked cortical potentials (VECP) that accompanied microsaccades. The results of his experiments showed a

linear relationship between microsaccade length and the level of visual response. These findings suggest that visual response was, at least in part, produced by sweeping visual stimuli across several receptors using microsaccadic movement. Armington introduced the concept of additivity when interpreting his results, suggesting that electrical activity measured in the VECF was produced by stimulating separate receptor units and adding the responses to produce a total response. He used this concept, along with saccade arc length and cone size to calculate an approximate reaction potential on a per receptor basis. Armington's observations and conclusions are in accordance with the other work presented thus far, which supports the idea of a useful, unique purpose for microsaccades in the human visual system.

The final supporting documentation also lends credence to the idea that microsaccades may help to compensate for inoperative or insensitive receptors, by passing visual stimuli over several receptors. Zihl [5] trained patients with varying degrees of cerebral blindness caused by retinal scotoma to make controlled saccadic movements, and in doing so, increased the size of their visual fields. The majority of participating patients had had a visual field defect for less than one year, and no patient had experienced any spontaneous recovery of visual field during that time. Through Zihl's training, the patients learned to make saccades in the affected area, and to increase their visual fields. Zihl found that the success of the therapy was significantly greater when the gradient of light sensitivity between the healthy area and the scotoma was shallow, than when it was steep. He also found that light stimulus was critical for visual field enlargement, and that with progress, the gradient of light sensitivity between the healthy and affected areas became

steeper [5]. These findings are significant because they seem to indicate a direct causal relationship between saccadic movement and increased visual response, in areas with a scattering of inoperative receptors. These results do not extend to microsaccadic movement; they are described here because they support the project hypothesis directly, and because there is no evidence available to suggest that similar therapeutic results could not be obtained using microsaccadic movement.

There is documentation available which maintains that microsaccades serve no useful purpose in the human visual system. The primary criterion for this assumption is that microsaccades decrease in frequency during observational tasks requiring high visual acuity [1]. However, the decrease in microsaccades during high-acuity tasks is expected, if the control system modelled by Van Gisbergen is valid. Following the model, it is logical for microsaccades to decrease as the oculomotor error approaches zero, that is, just as the eye fixates directly on its target [7]. It is apparent that those researchers who have concluded that microsaccades are useless have done so because they have been unable to determine a valid purpose for them, that is, a purpose that is not shared with the larger saccades.

The work which does not support the original hypothesis is not to be discounted. However, there exists ample support within this report of the usefulness of microsaccadic movement in the human visual system. The direct relationship between saccades and microsaccades in terms of origin, velocity-amplitude characteristics, and amplitude-response characteristics has been established. A unique purpose of microsaccades, that of sending visual information to the brain during periods of receptor insensitivity (binocular

suppression) has been suggested and supported. A direct relationship between the number of receptors activated in a microsaccadic movement and the level of visual response evoked has been established. Finally, evidence that controlled saccadic movements increase the visual fields of patients with areas of inoperative receptors has been produced. Although none of the documentation presented here may be considered conclusive, in terms of validating the original hypothesis, as a cohesive unit, the work presented provides a sound basis for the modelling of emulated microsaccadic activity in an electronic detection system, and for testing the hypothesis with original research.

REFERENCES

1. B. Winterson and H. Collewyn, "Microsaccades During Finely Guided Visuomotor Tasks," *Vision Research*, vol. 16, pp. 1387-1390, 1976.
2. R. M. Steinman, R. J. Cunitz, and G. T. Timberlake, "Voluntary Control of Microsaccades During Maintained Monocular Fixation," *Science*, vol. 155, pp. 1577-1579, 1967.
3. G. M. Haddad and R. M. Steinman, "The Smallest Voluntary Saccade: Implications for Fixation," *Vision Research*, vol. 13, no. 1075-1086, 1972.
4. B. Bridgeman and J. Palca, "The Role of Microsaccades in High Acuity Observational Tasks," *Vision Research*, vol. 20, pp. 813-817, 1980.
5. J. Zihl, "Recovery of Visual Functions in Patients with Cerebral Blindness," *Experimental Brain Research*, vol. 44, pp. 159-169, 1981.
6. B. L. Zuber and L. Stark, "Microsaccades and the Velocity-Amplitude Relationship for Saccadic Eye Movement," *Science*, vol. 150, pp. 1459-1460, 1965.
7. J. A. M. Van Gisbergen, D. A. Robinson, and S. Gielen, "A Qualitative Analysis of Generation of Saccadic Eye Movements by Burst Neurons," *Journal of Neurophysiology*, vol. 45, no. 3, pp. 417-422, 1981.
8. R. L. Gregory, *Eye and Brain: The Psychology of Seeing*, McGraw-Hill, New York, NY, 1977, p. 56.
9. H. W. Sabrin and A. E. Kertesz, "The Effect of Imposed Fixational Eye Movements on Binocular Rivalry," *Perception and Psychophysics*, vol. 34, pp. 155-157, 1983.
10. H. W. Sabrin and A. E. Kertesz, "Microsaccadic Eye Movements and Binocular Rivalry," *Perception and Psychophysics*, vol. 28, pp. 150-154, 1980.
11. G. W. Beeler, "Visual Threshold Changes Resulting from Spontaneous Saccadic Eye Movements," *Vision Research*, vol. 7, pp. 769-775, 1967.

12. J. C. Armington and M. B. Bloom, "Relations Between the Amplitudes of Spontaneous Saccades and Visual Responses," *Journal of the Optical Society of America*, vol. 64, no. 9, pp. 1263-1271, 1974.

APPENDIX A.1

LIST OF PROGRAMS USED IN CHAPTER I

All processing was done on a VAX 11-750 minicomputer with an AP-10A-1 Avalon attached processor, a Colorado Video frame grabber and digitizer and a Lexidata color graphics system.

```

c *****
c *
c *   Program to generate Leroy File to plot sensor design
c *   consisting of concentric circles and radial lines,
c *   which gives the distorted rectangle element shape.
c *
c *****

```

```

Real r,s,t,u,v,e,f,l
real radius(100), thetad(100)
integer a, b

```

```

c   First enter the size of the array

```

```

c   Number of circles(rings)

```

```

print*, 'Enter the number of rings'
read*, m

```

```

c   Number of lines

```

```

print*, 'Enter the number of elements per ring'
read*, n

```

```

c   Exponential spacing

```

```

print*, 'Enter exponential spacing constant'
read*, e
r=e**m

```

```

c   Now must create the Leroy File

```

```

write(2,21)
write(1,21)
21 format('.clear')
u=-1.6*r
v=1.6*r
t=-1.2*r
s=1.2*r
write(1,22) u,t,v,s
22 format('.map ('',f10.2,'',f10.2,') ('',f10.2,'',f10.2,')')
write(2,23)
23 format('.map (-0.6,-0.2) (1.6,1.2)')
write(2,24)
24 format('.draw (0,0) (1,0) {thick=0.03}')
write(2,27)
27 format('.draw (0,0) (0,1) {thick=0.03}')
do 10 a=1,n
    thetad(a)=(a*360/n)

```

```

        write(1,25) r,thetad(a)
25      format('.lline (0,0) length=',f10.2,' angle=',f10.2)
        l=real(a)/real(n)
        write(2,26) l,l
26      format('.draw (0,',f10.8,') (1,f10.8)')
10      continue
        do 20 b=1,m
            radius(b)= e**b
            write(1,35) radius(b)
35      format('.circle (0,0) radius=',f10.2,' narcs=100')
            f=real(b)/real(m)
            write(2,36) f,f
36      format('.draw (' ,f10.8,',0) (' ,f10.8,',1)')
20      continue
        write(2,40)
40      format('.to (1.5,1.1) cmode=8')
        write(1,41) r,r
41      format('.to (' ,f10.2,',',f10.2,') cmode=8')
        write(1,42) m,n,e
        write(2,42) m,n,e
42      format(i2,',',i2,',',f4.2)

c      This file is now readable by Leroy
      end

```



```

c *****
c ****
c ****      Program to generate the log spiral vision sensor
c ****      structure for circular elements.
c ****
c *****

```

```

integer n,m,o
real a, alpha(200), radius(100)
real pi, k, max, r(100),x ,y,l
real lbx, lby, ubx, uby
real lmx,lmy,umx,umy
real z1,z2
real rad

```

```

pi=3.1416

```

c Set variables from input.

```

print*,'Enter the number of elements per ring.'
read*,n
print*,'Enter the number of rings'
read*,m
print*,'Enter percentage overlap'
read*,o

```

c Calculate angular spacing.

```

a=real(pi/n)

do 1 i=1,2*n
    alpha(i)=real(i*a)
1  continue

call finder(n,rad)

```

c Calculate the spacing of the rings.

```

l=.01*o

k=rad

do 2 i=1,m
    r(i)=k**i
    radius(i)=((1+l)**.5)*(k**i)*sin(a)
2  continue

max=r(m)

```

c Create leroy file.

```
write(2,10)
write(1,10)
10 format('clear')
```

c Set proper map size.

```
lhx=-1.6*max
ubx=1.6*max
lhy=-1.2*max
uby=1.2*max
lmx=-.6*1.732*(m-1)+2
umx=1.6*1.732*(m-1)+2
lmy=-.2*2*n
umy=1.2*2*n

write(2,11) lmx,lmy,umx,umy
write(1,11) lhx,lhy,ubx,uby
11 format('map ('f10.2,',',f10.2,') ('f10.2,',',f10.2,')')

13 format('circle ('f10.2,',',f10.2,') radius=',f10.2,'
& narcs=100')
```

c Calculate coordinates of centers.

```
do 5 i=1,2*n
  if((real(i)/2).eq.(int(real(i)/2))) then
    do 3 j=1,m/2
      x=r(2*j)*cos(alpha(i))
      y=r(2*j)*sin(alpha(i))
      write(1,13) x,y,radius(2*j)
3    continue
  else
    do 4 j=1,m/2
      x=r(2*j-1)*cos(alpha(i))
      y=r(2*j-1)*sin(alpha(i))
      write(1,13) x,y,radius(2*j-1)
4    continue
  end if
5 continue

do 20 i=1,n
  do 21 j=1,m/2
    x=1.732*(2*j-1)+1
    y=2*i
    write(2,13) x,y,(1+1)**.5
```

```

21      continue
      do 22 j=1,m/2
          x=1.732*(2*j-2)+1
          y=2*i-1
          write(2,13) x,y,(1+i)**.5
22      continue
20  continue

      z1=1.5*1.732*(m-1)+2
      z2=1.1*2*n

      write(2,14) z1,z2
      write(1,14) max,max
14  format('to ('f10.2,',',f10.2,') cmode=8')
      write(2,15) n,m,o
      write(1,15) n,m,o
15  format(i2,',',i2,',',i2)

      write(2,16) 1.732*(m-1)+2
16  format('draw (0,0) ('f10.2,',0) {thick=0.03}')
      write(2,17) real(2*n)
17  format('draw (0,0) (0,f10.2,') {thick=0.03}')

      end

```

```

c *****
c ****
c ****      SUBROUTINE TO FIND THE ROOTS FOR THE CIRCULAR
c ****      ELEMENT CASE
c ****
c *****

```

```

subroutine finder(n,rad)

```

```

integer      n,iflag
real         alpha,pi,a,b,xtol,error,fa,fm,xm
real         rad

```

```

pi=3.141592
alpha=real(pi/n)

```

```

c  f(x)=alpha-x+asin(1-sin(x))

```

```

a=pi/6
b=pi/4

```

```

xtol=.00001

```

```

fa=alpha-a+asin(1-sin(a))
fb=alpha-b+asin(1-sin(b))
if (fa*fb.gt.0.) then

```

```

        iflag=-1
        print 601,a,b
601      format('SAME SIGN AT 2 ENDPOINTS',2f15.7)
        goto 100
    end if

```

```

error=abs(b-a)

```

```

6  error=error/2
    if (error.lt.xtol) then
        goto 100
    end if
    xm=(a+b)/2
    if (xm+error.eq.xm) then
        iflag=1
        print 602,a,b
602      format('LIMIT REACHED',2f15.7)
        goto 100
    end if
    fm=alpha-xm+asin(1-sin(xm))
    if (fa*fm.gt.0.) then

```

```
        a=xm
        fa=fm
    else
        b=xm
    end if
    goto 6
100 print*,'OK SO FAR',fa,fb,fm,a,b,xm

    rad=sin(a)/(1-sin(a))
    print*,rad

    return

end
```

```

c *****
c **** PROGRAM TO SIMULATE THE LOG SPIRAL STRUCTURE
c **** CONSISTING OF RECTANGULAR ELEMENTS, ALSO
c **** CAN BE THOUGHT OF AS CONCENTRIC CIRCLES AND
c **** RADIAL LINES.
c *****

```

c DECLARATIONS

```

integer      n, m
real         e, maxrad, scale
real         radius(100)
integer      nrows, ncols, file
integer*2    temp(-256:256,-256:256)
integer      p,stacki1(5000),stackj1(5000)
real         numave1,numave2,numave3,numave4,slope(100)
integer      ave1,ave2,ave3,ave4
integer      xmax,xmin,ymax,ymin
integer*2    comp(100,100)
integer      v
common       n,m,temp,comp

```

c USER SELECTABLE SENSOR CHARACTERISTICS m,n,e

```

print*, 'Enter the desired number of elements per ring'
read*, n

print*, 'Enter the desired number of rings'
read*, m
m=m+1

print*, 'Enter the desired exponential spacing constant'
read*, e

maxrad = e**m
scale = 256/maxrad

```

c NOW MUST CALCULATE THE PROPER RADII.

```

do 1 i=1,m

      radius(i) = scale*(e**i)

1  continue

```

```

do 100 i=1,n/4
    slope(i)=tan((i-1)*2*3.1416/n)
100 continue
c NOW TO CREATE THE IMAGE.
call rdhead(nrows,ncols,file)
do 2 i=-256,ncols-256
    call rdcol(temp(-256,i),nrows, file)
2    continue
    call closef(file)
c MASK OFF THE FOVEA AND OUTER PERIPHERY.
call mask( radius(m), radius(1) )
c NOW WE CREATE THE PIXEL FITTED SIMULATION.
do 20 i=1,m-1
    do 10 j=1,n/4
        if (j.eq.(n/4)) then
            v=i
            call last( radius(i), radius(i+1), slope(j), v )
            goto 20
        end if
        call iso( radius(i), radius(i+1), slope(j), slope(j+1),
&                xmax, xmin, ymax, ymin )

        p=0
        numave1=0
        numave2=0
        numave3=0
        numave4=0
        ave1=0
        ave2=0
        ave3=0

```

```

ave4=0

do 50 k=ymax,ymin,-1
    do 40 l=xmin,xmax

        if (((k**2+l**2).gt.(radius(i)**2)).and.
            &      ((k**2+l**2).lt.(radius(i+1)**2))) then

            if (((k).gt.(1*slope(j))).and.
                &      ((k).lt.(1*slope(j+1)))) then

                p=p+1

                stackil(p)=k
                stackjl(p)=l

                numave1=numave1+temp(k,l)
                numave2=numave2+temp(k,-l)
                numave3=numave3+temp(-k,-l)
                numave4=numave4+temp(-k,l)

            end if

        end if

40      continue

50      continue

    if (p.eq.0) then

        print*, 'Stopped due to p=0 at', i, j
        print*, xmax, xmin, ymax, ymin
        goto 10

    end if

    ave1= int(numave1/p)
    ave2= int(numave2/p)
    ave3= int(numave3/p)
    ave4= int(numave4/p)

    comp(j,i)=ave1
    comp(n/2-(j-1),i)=ave2
    comp(j+n/2,i)=ave3
    comp(n-(j-1),i)=ave4

```



```

do 60 k=p,1,-1
    temp(stacki1(k),stackj1(k))=ave1
    temp(stacki1(k),-stackj1(k))=ave2
    temp(-stacki1(k),-stackj1(k))=ave3
    temp(-stacki1(k),stackj1(k))=ave4
60      continue
10      continue
20      continue

```

c NOW TO WRITE THE FILE BACK

```

    call wrhead(nrows,ncols,file)
    do 5 i=-256,ncols-256
        call wrcol(temp(-256,i),nrows, file)
5      continue

```

c NOW TO CREATE THE MAPPED IMAGE.

```

    call spread
    end

```

```
c *****
c **** SUBROUTINE TO ISOLATE ONE POSSIBLE ELEMENT'S
c **** PIXELS TO AVOID MASSIVE AMOUNTS OF WASTED
c **** COMPUTER TIME.
c *****
```

```
subroutine iso(r1,r2,m1,m2,xmax,xmin,ymax,ymin)
```

```
integer      xmax,xmin,ymax,ymin
real         r1,r2,m1,m2,x1,x2,x1a,x2a
real         y1,y2,y1a,y2a
```

```
x1=r1/((1+m1**2)**.5)
x2=r2/((1+m1**2)**.5)
x1a=r1/((1+m2**2)**.5)
x2a=r2/((1+m2**2)**.5)
```

```
y1=x1*m1
y2=x2*m1
y1a=x1a*m2
y2a=x2a*m2
```

```
xmax=int(max(x1,x2,x1a,x2a))
xmin=int(min(x1,x2,x1a,x2a))
ymax=int(max(y1,y2,y1a,y2a))
ymin=int(min(y1,y2,y1a,y2a))
```

```
return
```

```
end
```

```

C *****
C **** SUBROUTINE TO CALCULATE THE PROPER STUFF FOR THE
C **** LAST ELEMENT IN EACH QUADRANT IN EACH ROW
C *****

```

```

      subroutine last( r1, r2, slope, v )

```

```

      integer*2      temp(-256:256,-256:256)
      real           r1, r2, slope
      real           numave1, numave2, numave3, numave4
      integer        xmax, xmin, ymax, ymin
      integer        ave1, ave2, ave3, ave4, p
      integer        stacki(5000),stackj(5000)
      integer        n, v, a, b
      integer*2      comp(100,100)
      common         n,m,temp,comp

```

```

      ymax= int(r2)
      ymin= int(r1/((1+slope**2)**.5))
      xmin= 0
      xmax= int(r2/((1+slope**2)**.5))

```

```

      p=0
      numave1=0
      numave2=0
      numave3=0
      numave4=0
      ave1=0
      ave2=0
      ave3=0
      ave4=0

```

```

      do 2 i=ymin,ymax

```

```

          do 1 j=xmin,xmax

```

```

              if (((i**2+j**2).gt.(r1**2)).and.
%              ((i**2+j**2).lt.(r2**2))) then

```

```

                  if (i.gt.(j*m1)) then

```

```

                      p=p+1

```

```

                      stacki(p)=i
                      stackj(p)=j

```

```

                      numave1=numave1+temp(i,j)
                      numave2=numave2+temp(i,-j)

```

```
numave3=numave3+temp(-i,-j)
numave4=numave4+temp(-i,j)
```

```
end if
```

```
end if
```

```
1      continue
```

```
2      continue
```

```
ave1=int(numave1/p)
ave2=int(numave2/p)
ave3=int(numave3/p)
ave4=int(numave4/p)
```

```
a=int(n/4)
b=int(3*n/4)
```

```
comp(a,v)=ave1
comp(a+1,v)=ave2
comp(b,v)=ave3
comp(b+1,v)=ave4
```

```
do 3 i=p,1,-1
```

```
temp(stacki(i),stackj(i))=ave1
temp(stacki(i),-stackj(i))=ave2
temp(-stacki(i),-stackj(i))=ave3
temp(-stacki(i),stackj(i))=ave4
```

```
3      continue
```

```
return
```

```
end
```

```

c *****
c **** SUBROUTINE TO MASK OFF THE FOVEA AND PERIPHERY
c **** OF THE IMAGE THAT WILL NOT BE USED.
c *****

```

```

subroutine mask( rmax, rmin)

```

```

integer*2    temp(-256:256,-256:256)
real         rmax,rmin
integer*2    comp(100,100)
common       n,m,temp,comp

```

```

do 2 i=0,256

```

```

do 1 j=0,256

```

```

&         if (((i**2+j**2).ge.(rmax**2)).or.((i**2+j**2).le.
           (rmin**2))) then

```

```

           temp(i,j)=0
           temp(i,-j)=0
           temp(-i,j)=0
           temp(-i,-j)=0

```

```

           end if

```

```

1         continue

```

```

2         continue

```

```

return

```

```

end

```

```

C *****
C **** SUBROUTINE TO CREATE THE COMPUTATIONAL PLANE
C **** MAPPING OF THE PIXEL FITTED IMAGE IN THE
C **** IMAGE PLANE.
C *****

```

```

subroutine spread

integer*2    comp(100,100)
integer*2    scomp(512,512)
integer      n,m,ncols,nrows,file,a,b
integer*2    temp(-256:256,-256:256)
common       n,m,temp,comp

ncols=512
nrows=512

a=int(512/n)
b=int(512/m)

do 4 i=1,m
    do 3 j=1,n
        do 2 k=(i-1)*b,i*b
            do 1 l=512-(j)*a,512-(j-1)*a
                scomp(l,k)=comp(j,i)
1                continue
2            continue
3        continue
4    continue

call wrhead( nrows, ncols, file )

do 5 i=1,ncols
    call wrcol( scomp(1,i), nrows, file )
5    continue

return

end

```

```

c *****
c ****   PROGRAM TO SIMULATE THE LOG SPIRAL SENSOR
c ****   CONSISTING OF CIRCULAR ELEMENTS
c *****

```

```

integer*2    temp(-256:256,-256:256)
integer*2    comp(100,100)
integer      n,m,nrows,ncols,file,k,l,aveval,p
real         rmax,rmin,alpha,rad,ave,angle,r,radius
real         scale,maxrad,offset
integer      xstack(15000),ystack(15000),x,y
integer      xmax,xmin,ymax,ymin
common       n,m,temp,comp

```

```

print*,'ENTER THE DESIRED NUMBER OF ELEMENTS PER RING'
read*,n

```

```

print*,'ENTER THE DESIRED NUMBER OF RINGS'
read*,m

```

```

call rdhead(nrows,ncols,file)

```

```

do 1 i=-256,ncols-256
    call rdcol(temp(-256,i),nrows,file)
1 continue

```

```

call closefl(file)

```

```

alpha=real(3.141592/n)

```

```

call finder(n,rad)

```

```

maxrad=(rad**m)*(1+sin(alpha))

```

```

scale=256./maxrad

```

```

rmax=256.

```

```

rmin=scale*(1-sin(alpha))

```

```

call mask( rmax,rmin )

```

```

do 5 i=1,m

```

```

    radius=scale*(rad**i)
    r=radius*sin(alpha)
    offset=0

```

```

if ((int(real(i)/2)).eq.(real(i)/2)) then
    offset=alpha
end if
do 4 j=1,n
    angle=2*j*alpha+offset
    call isol(radius,angle,r,xmax,xmin,ymax,ymin,x,y)

    p=0
    ave=0
    aveval=0

    do 3 k=xmin,xmax
        do 2 l=ymin,ymax
            if (((k-x)**2)+((l-y)**2)).le.(r**2)) then

                p=p+1
                xstack(p)=k
                ystack(p)=l
                ave=ave + temp(k,l)

            end if
        2      continue
    3      continue

    if (p.eq.0) then
        print*,'stopped due to p=0 at',i,j
        print*,x,y,xmax,xmin,ymax,ymin
        goto 4
    end if

    aveval=int(ave/p)
    comp(j,i)=aveval

    do 10 k=1,p
        temp(xstack(k),ystack(k))=aveval

```



```

10          continue
4          continue
5  continue
    call wrhead(n,m,file)
    do 100 i=1,m
        call wrcol(comp(1,i),n,file)
100 continue
    call send
    call map()
    call wrhead(nrows,ncols,file)
    do 6 i=-256,ncols-256
        call wrcol(temp(-256,i),nrows,file)
6  continue
    end

```

```

C *****
C ****
C ****   SUBROUTINE TO ISOLATE THE ELEMENTS
C ****   IN THE CIRCULAR ELEMENT CASE
C ****
C *****

```

```

      subroutine isol( radius,angle,r,xmax,xmin,ymax,ymin,x,y )

```

```

      real  r,angle,radius,xt,yt
      integerxmax,xmin,ymax,ymin,x,y

```

```

      xt=radius*cos((3.141592/2)+angle)
      yt=radius*sin((3.141592/2)+angle)

```

```

      x=int(xt)
      y=int(yt)

```

```

      xmax=int(xt+r)
      xmin=int(xt-r)
      ymax=int(yt+r)
      ymin=int(yt-r)

```

```

      return

```

```

      end

```

```

C *****
C ****
C ****      SUBROUTINE TO MAP THE SIMULATION INTO THE
C ****      COMPUTATIONAL PLANE FOR THE CIRCULAR
C ****      ELEMENT CASE
C ****
C *****

      subroutine map()

      integer*2    temp(-256:256,-256:256)
      integer*2    comp(100,100)
      integer      n,m,nrows,ncols,file
      integer      k,l,r,x,y,offset
      integer*2    maplane(512,512)
      common       n,m,temp,comp

      nrows=512
      ncols=512

      r=int(min((512/(2*n+1)),(512/(2+1.73205*(m-1)))))

      do 4 i=1,m
         offset=0
         if ((int(real(i)/2)).eq.(real(i)/2)) then
            offset=r
         end if
         y=r+int((i-1)*1.73205*r)
         do 3 j=1,n
            x=513-(r+(j-1)*2*r+offset)
            do 2 k=x-r,x+r
               do 1 l=y-r,y+r
                  if (((k-x)**2)+((l-y)**2)).le.(r**2)) then
                     maplane(k,l)=comp(j,i)
                  end if
               end do
            end do
         end do
      end do

```

```
1             continue
2             continue
3             continue
4             continue
              call wrhead(nrows,ncols,file)
              do 7 i=1,ncols
                  call wrcol(maplane(1,i),nrows,file)
7             continue
              return
              end
```

```

c *****
c **** SUBROUTINE TO MASK OFF THE FOVEA AND PERIPHERY
c **** OF THE IMAGE THAT WILL NOT BE USED.
c *****

```

```

subroutine mask( rmax, rmin)

```

```

integer*2    temp(-256:256,-256:256)
real         rmax,rmin
integer*2    comp(100,100)
common       n,m,temp,comp

```

```

do 2 i=0,256

```

```

do 1 j=0,256

```

```

&      if (((i**2+j**2).ge.(rmax**2)).or.((i**2+j**2).le.
      (rmin**2))) then

```

```

      temp(i,j)=0
      temp(i,-j)=0
      temp(-i,j)=0
      temp(-i,-j)=0

```

```

      end if

```

```

1      continue
2      continue

```

```

return

```

```

end

```

```

c *****
c ****
c ****      SUBROUTINE TO FIND THE ROOTS FOR THE CIRCULAR
c ****      ELEMENT CASE
c ****
c *****

```

```

      subroutine finder(n,rad)

      integer      n,iflag
      real         alpha,pi,a,b,xtol,error,fa,fm,xm
      real         rad

      pi=3.141592
      alpha=real(pi/n)

c      f(x)=alpha-x+asin(1-sin(x))

      a=pi/6
      b=pi/4

      xtol=.00001

      fa=alpha-a+asin(1-sin(a))
      fb=alpha-b+asin(1-sin(b))
      if (fa*fb.gt.0.) then
         iflag=-1
         print 601,a,b
         format('SAME SIGN AT 2 ENDPOINTS',2f15.7)
         goto 100
      end if

      error=abs(b-a)

      6      error=error/2
      if (error.lt.xtol) then
         goto 100
      end if
      xm=(a+b)/2
      if (xm+error.eq.xm) then
         iflag=1
         print 602,a,b
         format('LIMIT REACHED',2f15.7)
         goto 100
      end if
      fm=alpha-xm+asin(1-sin(xm))
      if (fa*fm.gt.0.) then

```

```
        a=xm
        fa=fm
    else
        b=xm
    end if
    goto 6
100 print*,'OK SO FAR',fa,fb,fm,a,b,xm

    rad=sin(a)/(1-sin(a))
    print*,rad

    return

end
```

```

C *****
C *****
C ***** SUBROUTINE TO WRITE OUT MAPPING
C ***** INFORMATION TO BE USED WITH
C ***** EDGE DETECTION OPERATOR
C *****
C *****

```

```

subroutine send

```

```

integer*2    temp(-256:256,-256:256)
integer*2    comp(100,100)
integer      n,m

```

```

common      n,m,temp,comp

```

```

write(1,10) n
write(1,10) m

```

```

do 2 i=1,m

```

```

    do 1 j=1,n

```

```

        write(1,10) comp(j,i)

```

```

1          continue

```

```

2          continue

```

```

10         format(i10)

```

```

return

```

```

end

```


APPENDIX A.2

LIST OF PROGRAMS USED IN CHAPTER II

program hedge

c This program simulates edge detection for human PVS.
c It generates the edge data using Laplacian-Gaussian
c function, and gives an average value for every sampling point.

```
integer*2      f(512,512)
integer*2      ximg(150,250),xedge(150,250)
integer*2      lm(300),ln(300)
integer*2      ir(150)
integer        rows,cols
real           rz(150)
character*40    filename

common lm,ln
common /b1/ f
common /b2/ ximg,xedge

open(3,file='testdata')
rows=512
cols=512
call num1
call num2
5  write(6,10)
   write(6,20)
   write(6,30)
   write(6,40)
   write(6,50)
   write(6,60)
   write(6,70)
   write(6,80)
10  format(/,'Your options are: ')
20  format(' 1=Create an image')
30  format(' 2=Input an image from camera')
40  format(' 3=Input the parameters for the RF of PV')
50  format(' 4=Image operation')
60  format(' 5=Edge operation')
70  format(' 6=quit')
80  format(/,'Selection? ',)$

read(5,*) imenu

go to(100,200,300,400,500,9999) imenu
go to 5

100 call square(rows,cols)
    go to 5
200 call inputim
    go to 5
300 call paramet(ir,rz,nr1,nr2,ne,thetar)
    go to 5
400 write(6,410)
410 format(/,'What is the name of file: ',)$
```

```

        read(5,*)filename
        open(1,file=filename)
        call imgop(ir,rz,nr1,ne,thetar)
        go to 5
500    write(6,510)
510    format(//,'What is the name of file: ',%)
        read(5,*)filename
        open(2,file=filename)
        call edgeop(ir,rz,nr2,ne,thetar)
        go to 5
9999   stop
        end

```

```

        subroutine square(rows,cols)
c      Creat an binary image.

        integer*2 f(512,512)
        integer   rows, cols
        common /b1/ f

        do 1 i=1,rows
          do 2 j=1,cols
            if(i .ge. 150 .and. i .le. 350) then
              if(j .ge. 150 .and. j .le. 350) then
                f(i,j)=200
              else
                f(i,j)=125
              endif
            else
              f(i,j)=125
            endif
          2   continue
        1   continue
        return
        end

```

```

        subroutine inputim
c      Input an image from a camera.

        integer*2 f(512,512)
        integer*2 col(512)
        integer fp

```

```

common /b1/ f

call rdhead(nrow,ncol,fp)
do 1 j=1,ncol
  call rdcol(col,nrow,fp)
  do 2 i=1,nrow
    f(i,j)=col(i)
2    continue
1    continue
return
end

```

```

subroutine paramet(ir,rz,nr1,nr2,ne,thetar)

```

c It gives the number of elements per ring, the number
c of rings and their coordinates.

```

integer*2 ir(150)
real rz(150)
integer w1, w

write(6,1)
1 format(/,'Please input the smallest diameter (integer value) of the
+ central region of the overlapping receptive field, w1')
read(5,*) w1
write(6,2) w1
2 format(1x,'w1= ',i1)
write(6,3)
3 format(/,'Please input the smallest integer value of the eccentric
+ ity, ir(1)')
read(5,*) ir(1)
write(6,4) ir(1)
4 format(1x,'ir(1)= ',i2)
write(6,5)
5 format(/,'Please input the overlapping factor, of')
read(5,*) of
write(6,6) of
6 format(1x,'of= ',f10.5)
pi=3.14159
r=float(ir(1))
rz(1)=w1/2.
xrz=rz(1)
rzn=(1.-of)*rz(1)
rn=pi/asin(rzn/r)
ne=int(rn+0.5)
thetar=pi/ne
a=(tan(thetar))**2.+1./cos(thetar)
i=1
9 if(((ir(i)+rz(i)) .le. 256.) then

```

```

        if((ir(i)+4.5*rz(i)) .le. 256.) then
            nr2=i
        endif
        write(6,10) i, rz(i),xrz, i, ir(i),r
10      format(/,'Radius rz('i2,')='f7.2,2x,f7.2,5x,'Eccen. ir('i2,')=',
+      i3,2x,f8.2)
        r=r*(a+(a**2.-1.)*0.5)
        xrz=(r*sin(thetar))/(1.-of)
        w=int(xrz*2.+0.5)
        i=i+1
        ir(i)=int(r+0.5)
        rz(i)=w/2.
        go to 9
    endif
    nr1=i-1
    write(6,7) ne
7    format(/,'Number of elements per ring, ne= ', i3)
    write(6,8) thetar
8    format(/,'thetar= ',f10.5)
    write(6,11) nr1
11   format(/,'Number of rings for image, nr1= ',i2)
    write(6,12) nr2
12   format(/,'Number of rings for edge, nr2= ',i2)
    end

```

```

subroutine imgop(ir,rz,nr,ne,thetar)

```

```

c      It generates an average value at every sampling point
c      for the original image.

```

```

    integer*2  f(512,512)
    integer*2  ximg(150,250), xedge(150,250)
    integer*2  lm(300),ln(300)
    integer*2  ir(150)
    real       rz(150)

    common lm,ln
    common /b1/ f
    common /b2/ ximg, xedge
    do 1 i=1,nr
        id=int(rz(i)+0.5)-int(rz(i))
        if( id .eq. 0) then
            call ifitte(i,ir(i),rz(i),ne,thetar)
        else
            call ifitto(i,ir(i),rz(i),ne,thetar)
        endif
1    continue
    return
end

```

subroutine edgeop(ir,rz,nr,ne,thetar)

c It generates the edge data using the Laplacian-Gaussian
c function.

```
integer*2 f(512,512)
integer*2 ximg(150,250), xedge(150,250)
integer*2 lm(300), ln(300)
integer*2 ir(150)
real      rz(150)
real      w(300)
```

```
common lm, ln
common /b1/ f
common /b2/ ximg, xedge
```

```
thetae=2.*thetar
do 3 i=1,nr
  p=rz(i)**2.
  irz=int(rz(i)*4.5)+1
  nw=irz+1
  w(1)=0.
  do 10 i1=2,nw
    w(i1)=((i1-1.5)**2.)*exp(-(i1-1.5)**2./p)
10  continue
  do 11 i2=2,nw
    w(i2-1)=w(i2)-w(i2-1)
11  continue
  do 4 j=1,ne
    pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir(i)*cos(pt)
    yp=256-ir(i)*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    t=f(nx,ny)*w(1)
    do 5 ik=2,irz
      s=0.
      d=ik-0.5
      m=ik-1
      n=0
      s=s+f(nx+m,ny)+f(nx-m,ny)+f(nx,ny+m)+f(nx,ny-m)
6      if(m .eq. 0) go to 8
7      n=n+1
      ds=m**2.+n**2.
      r2=d**2.
      r1=(d-1.)**2.
      if(ds .ge. r1) then
        if(ds .lt. r2) then
          s=s+f(nx+m,ny+n)+f(nx-m,ny+n)+f(nx+m,ny-n)
          +f(nx-m,ny-n)
          go to 7
```

```

        else
            m=m-1
            n=n-2
            go to 6
        endif
        else
            go to 7
        endif
8       t=t+s*w(ik)/lm(ik)
5       continue
        if(t .gt. -0.1 .and. t .lt. 0.1) then
            xedge(i,j)=0
        else
            if(t .gt. 0) then
                xedge(i,j)=int(t+1)
            else
                xedge(i,j)=int(t-1)
            endif
            write(2,*)xedge(i,j)
            write(3,*)i,j,t,xedge(i,j)
4       continue
3       continue
        return
    end

```

subroutine num1

c It gives the total number of pixels in a receptive field
c of odd-numbered diameter.

```

integer*2 lm(300), ln(300)

common lm, ln

lm(1)=1
write(6,2) lm(1)
2 format(1x,'lm( 1)= ',i3)
do 5 ik=2,100
    l=ik
    lm(1)=0
    d=ik-0.5
    m=ik-1
    n=0
    lm(1)=lm(1)+4
6    if(m .eq. 0) go to 8
7    n=n+1
    ds=m**2.+n**2.
    r2=d**2.

```

```

      r1=(d-1.)**2.
      if(ds .ge. r1) then
        if(ds .lt. r2) then
          lm(1)=lm(1)+4
          go to 7
        else
          m=m-1
          n=n-2
          go to 6
        endif
      else
        go to 7
      endif
8      write(6,1) 1, lm(1)
1      format(1x,'lm(',i3,')= ',i5)
5      continue
      return
      end

```

subroutine num2

c It gives the total number of pixels in a receptive field
 c of even-numbered diameter.

```

      integer*2 lm(300), ln(300)

      common lm, ln

      ln(1)=5
      write(6,2) ln(1)
2      format(1x,'ln( 1)= ',i3)
      do 5 ik=2,100
        l=ik
        ln(1)=0
        id=ik
        m=ik
        n=0
        ln(1)=ln(1)+4
6      if(m .eq. 0) go to 8
7      n=n+1
        mn=m**2+n**2
        nr2=id**2
        nr1=(id-1)**2
        if(mn .gt. nr1) then
          if(mn .le. nr2) then
            ln(1)=ln(1)+4
            go to 7
          else
            m=m-1

```



```

        n=n-2
        go to 6
    endif
    else
        go to 7
    endif
8   write(6,1) 1, ln(1)
1   format(1x,'ln(',i3,')= ',i5)
5   continue
    return
end

```

```

subroutine ifitte(i,ir,rz,ne,thetar)

```

```

c   It gives an average value for every element
c   of even-numbered diameter.

```

```

integer*2 f(512,512)
integer*2 ximg(150,250), xedge(150,250)
integer*2 lm(300),ln(300)
integer*2 ir

```

```

common lm,ln
common /b1/ f
common /b2/ ximg, xedge

```

```

thetae=2.*thetar
irz=int(rz)
do 40 j=1,ne
    pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir*cos(pt)
    yp=256-ir*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    it=f(nx,ny)+f(nx+1,ny)+f(nx-1,ny)+f(nx,ny+1)+f(nx,ny-1)
    ft=float(it)
    nt=ln(1)
    do 50 ik=2,irz
        s=0.
        nt=nt+ln(ik)
        id=ik
        m=ik
        n=0
        s=s+f(nx+m,ny)+f(nx-m,ny)+f(nx,ny+m)+f(nx,ny-m)
60      if(m .eq. 0) go to 80
70      n=n+1
        mn=m**2+n**2
        nr2=id**2
        nr1=(id-1)**2
    enddo
enddo

```

```

        if(mn .gt. nr1) then
            if(mn .le. nr2) then
                s=s+f(nx+m,ny+n)+f(nx-m,ny+n)+f(nx+m,ny-n)
                +f(nx-m,ny-n)
            +
            go to 70
            else
                m=m-1
                n=n-2
            go to 60
        endif
        else
            go to 70
        endif
80      ft=ft+s
50      continue
        fav=ft/nt
        ximg(i,j)=int(fav+0.5)
        write(1,*)ximg(i,j)
40      continue
        return
        end

```

```

subroutine ifitto(i,ir,rz,ne,thetar)

```

```

c      It gives an average value for every element
c      of odd-numbered diameter.

```

```

integer*2 f(512,512)
integer*2 ximg(150,250), xedge(150,250)
integer*2 lm(300), ln(300)
integer*2 ir

```

```

common lm, ln
common /b1/ f
common /b2/ ximg, xedge

```

```

thetae=2.*thetar
irz=int(rz)+1
do 4 j=1,ne
    pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir*cos(pt)
    yp=256-ir*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    ft=float(f(nx,ny))
    nt=lm(1)
    do 5 ik=2,irz
        s=0.
        nt=nt+lm(ik)
    enddo
enddo

```

```

d=ik-0.5
m=ik-1
n=0
s=s+f(nx+m,ny)+f(nx-m,ny)+f(nx,ny+m)+f(nx,ny-m)
6   if(m .eq. 0) go to 8
7   n=n+1
    ds=m**2.+n**2.
    r2=d**2.
    r1=(d-1.)**2.
    if(ds .ge. r1) then
      if(ds .lt. r2) then
        s=s+f(nx+m,ny+n)+f(nx-m,ny+n)+f(nx+m,ny-n)
        +   f(nx-m,ny-n)
        go to 7
      else
        m=m-1
        n=n-2
        go to 6
      endif
    else
      go to 7
    endif
8   ft=ft+s
5   continue
    fav=ft/nt
    ximg(i,j)=int(fav+0.5)
    write(1,*)ximg(i,j)
4   continue
return
end

```

```

program hdispedge

c      This program displays images using the data
c      which is generated by the program of human PVS.

integer*2      h(512,512)
integer*2      ximg(150,250),xedge(150,250),x(150,250)
integer*2      ir(150)
integer        rows,cols,plane,threshold,level
real           rz(150)
logical        lexon,readimg,readedge
character*40    filename

common /b1/ h
common /b2/ ximg,xedge,x

readimg=.false.
readedge=.false.
lexon=.false.
rows=512
cols=512
call dparamet(ir,rz,nr1,nr2,ne,thetar)
5  write(6,10)
   write(6,20)
   write(6,30)
   write(6,40)
   write(6,50)
   write(6,60)
   write(6,70)
   write(6,80)
   write(6,90)
   write(6,110)
   write(6,120)
10  format(/,'Your options are: ')
20  format(' 1=Create an image')
30  format(' 2=Input an image from camera')
40  format(' 3=Input the parameters for the RF of PV')
50  format(' 4=Display the original image')
60  format(' 5=Display the circu'ar fitting image')
70  format(' 6=Display the operated image')
80  format(' 7=Display the edge')
90  format(' 8=Write the image to a file for printing')
110 format(' 9=quit')
120 format(/,'Selection? ',)$

read(5,*) imenu

go to(100,200,300,400,500,600,700,800,9999) imenu
go to 5

100 call square(rows,cols)
    go to 5
200 call inputim

```

```

go to 5
300 call paramet(ir,rz,nr1,nr2,ne,thetar)
go to 5
400 call display(rows,cols,lexon)
go to 5
500 if(readimg .eq. .false.) then
    write(6,510)
510    format(//,'Read the image data file, filename is: ',%)
    read(5,*)filename
    open(1,file=filename)
    call rdimg(nr1,ne)
    readimg=.true.
endif
call planepar(plane)
if(plane .eq. 0) then
    call fitting(ir,rz,nr1,ne,thetar)
else
    call compuplane(nr1,ne)
endif
call display(rows,cols,lexon)
go to 5
600 if(readedge .eq. .false.) then
    write(6,605)
605    format(//,'Read the edge data file, filename is: ',%)
    read(5,*)filename
    open(2,file=filename)
    call rdedge(nr2,ne)
    readedge=.true.
endif
do 620 i=1,nr2
    do 610 j=1,ne
        x(i,j)=xedge(i,j)+127
610    continue
620 continue
call planepar(plane)
if(plane .eq. 0) then
    do 640 i=1,512
        do 630 j=1,512
            h(i,j)=127
630    continue
640    continue
    call fovea(ir(1),0)
    call imgplane(ir,nr2,ne,thetar)
else
    call compuplane(nr2,ne)
endif
call display(rows,cols,lexon)
go to 5
700 if(readedge .eq. .false.) then
    write(6,705)
705    format(//,'Read the edge data file, filename is: ',%)
    read(5,*)filename
    open(2,file=filename)

```

```

        call rdedge(nr2,ne)
        readedge=.true.
    endif
    call zerocross(nr2,ne)
    write(6,710)
710  format(/,'Thresholding ? if Yes, Please type "1".')
    read(5,*)threshold
    if(threshold .eq. 1)then
        write(6,720)
720  format(1x,'Please input the thresholding level?')
        read(5,*)level
        call thresh(nr2,ne,level)
    endif
    call planepar(plane)
    if(plane .eq. 0) then
        do 740 i=1,512
            do 730 j=1,512
                h(i,j)=0
730          continue
740          continue
            call fovea(ir(1),255)
            call imgplane(ir,nr2,ne,thetar)
        else
            call compuplane(nr2,ne)
        endif
        call display(rows,cols,lexon)
        go to 5
800  call wrim(rows,cols)
        go to 5
9999 stop
    end

```

```

subroutine square(rows,cols)

```

```

c      Creat an image.

```

```

integer*2 h(512,512)
integer rows, cols
common /b1/ h

do 1 i=1,rows
    do 2 j=1,cols
        if(i .ge. 150 .and. i .le. 350) then
            if(j .ge. 150 .and. j .le. 350) then
                h(i,j)=200
            else
                h(i,j)=125
            endif
        else

```

```

        h(i,j)=125
    endif
2    continue
1    continue
    return
end

```

```

subroutine inputim

```

```

c    Input an image from a camera.

```

```

    integer*2 h(512,512)
    integer*2 col(512)
    integer fp

    common /b1/ h

    call rdhead(nrow,ncol,fp)
    do 1 j=1,ncol
        call rdcol(col,nrow,fp)
        do 2 i=1,nrow
            h(i,j)=col(i)
2        continue
1    continue
    return
end

```

```

subroutine display(Rows,Cols,lexon)

```

```

c    Display the image using Lexidata device.

```

```

    integer*2 h(512,512)
    integer Rows,Cols,x,y
    integer*2 Row(512)
    logical lexon

    common /b1/ h

    if (.not. lexon) then
        call opdisp(0,0,Rows,Cols)
        lexon=.true.
    endif
    do 1 x=1,Rows
        do 2 y=1,Cols

```

```

        if (h(x,y) .gt. 255) then
            Row(y)=255
        else
            if(h(x,y) .lt. 0) then
                Row(y)=0
            else
                Row(y)=h(x,y)
            endif
        endif
2      continue
      call dsput(Row,Cols)
1     continue
      return
      end

```

```

subroutine dparamet(ir,rz,nr1,nr2,ne,thetar)

```

c It gives the number of elements per ring, the number
c of rings and their coordinates for the default input value.

```

integer*2  ir(150)
real       rz(150)
integer    w1, w

w1=2
ir(1)=18
of=0.2
pi=3.14159
r=float(ir(1))
rz(1)=w1/2.
xrz=rz(1)
rzn=(1.-of)*rz(1)
rn=pi/asin(rzn/r)
ne=int(rn+0.5)
thetar=pi/ne
a=(tan(thetar))**2.+1./cos(thetar)
i=1
9  if((ir(i)+rz(i)) .le. 256.) then
    if((ir(i)+4.5*rz(i)) .le. 256.) then
        nr2=i
    endif
    write(6,10) i, rz(i),xrz, i, ir(i),r
10  format(/,'Radius  rz(',i2,')=',f7.2,2x,f7.2,5x,'Eccen.  ir(',i2,')=',
+      i3,2x,f8.2)
    r=r*(a+(a**2.-1.)**0.5)
    xrz=(r*sin(thetar))/(1.-of)
    w=int(xrz*2.+0.5)
    i=i+1
    ir(i)=int(r+0.5)

```



```

        rz(i)=w/2.
        go to 9
    endif
    nr1=i-1
    write(6,7) ne
7   format(/,'Number of elements per ring, ne= ', i3)
    write(6,8) thetar
8   format(/,'thetar= ',f10.5)
    write(6,11) nr1
11  format(/,'Number of rings for image, nr1= ',i2)
    write(6,12) nr2
12  format(/,'Number of rings for edge, nr2= ',i2)
    end

```

```

subroutine paramet(ir,rz,nr1,nr2,ne,thetar)

```

```

c      It gives the number of elements per ring, the number
c      of rings and their coordinates.

```

```

    integer*2  ir(150)
    real       rz(150)
    integer    w1, w

    write(6,1)
1   format(/,'Please input the smallest diameter (integer value) of the
+   central region of the overlapping receptive field, w1')
    read(5,*) w1
    write(6,2) w1
2   format(1x,'w1= ',i1)
    write(6,3)
3   format(/,'Please input the smallest integer value of the eccentric
+   ity, ir(1)')
    read(5,*) ir(1)
    write(6,4) ir(1)
4   format(1x,'ir(1)= ',i2)
    write(6,5)
5   format(/,'Please input the overlapping factor, of')
    read(5,*) of
    write(6,6) of
6   format(1x,'of= ',f10.5)
    pi=3.14159
    r=float(ir(1))
    rz(1)=w1/2.
    xrz=rz(1)
    rzn=(1.-of)*rz(1)
    rn=pi/asin(rzn/r)
    ne=int(rn+0.5)
    thetar=pi/ne
    a=(tan(thetar))*2.+1./cos(thetar)

```

```

i=1
9  if((ir(i)+rz(i)) .le. 256.) then
    if((ir(i)+4.5*rz(i)) .le. 256.) then
        nr2=i
    endif
    write(6,10) i, rz(i),xrz, i, ir(i),r
10  format(/,'Radius rz(',i2,')=',f7.2,2x,f7.2,5x,'Eccen. ir(',i2,')=',
+    i3,2x,f8.2)
    r=r*(a+(a**2.-1.)*0.5)
    xrz=(r*sin(thetar))/(1.-of)
    w=int(xrz*2.+0.5)
    i=i+1
    ir(i)=int(r+0.5)
    rz(i)=w/2.
    go to 9
endif
nr1=i-1
write(6,7) ne
7  format(/,'Number of elements per ring, ne= ', i3)
write(6,8) thetar
8  format(/,'thetar= ',f10.5)
write(6,11) nr1
11 format(/,'Number of rings for image, nr1= ',i2)
write(6,12) nr2
12 format(/,'Number of rings for edge, nr2= ',i2)
end

```

```

subroutine fitting(ir,rz,nr,ne,thetar)

```

c It gives an average value for every pixel in an element.

```

integer*2 h(512,512)
integer*2 ximg(150,250), xedge(150,250), x(150,250)
integer*2 ir(150)
real rz(150)

common /b1/ h
common /b2/ ximg, xedge, x

do 3 i=1,512
  do 2 j=1,512
    h(i,j)=0
2  continue
3  continue
do 1 i=1,nr
  id=int(rz(i)+0.5)-int(rz(i))
  if( id .eq. 0) then
    call fitte(i,ir(i),rz(i),ne,thetar)
  else

```

```

        call fitto(i,ir(i),rz(i),ne,thetar)
    endif
1  continue
    return
end

```

```

subroutine fitte(i,ir,rz,ne,thetar)

```

```

c      It gives an average value for every pixel in an element
c      of even-numbered diameter.

```

```

integer*2 h(512,512)
integer*2 ximg(150,250), xedge(150,250), x(150,250)
integer*2 ir

```

```

common /b1/ h
common /b2/ ximg, xedge, x

```

```

thetae=2.*thetar
irz=int(rz)
do 40 j=1,ne
    pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir*cos(pt)
    yp=256-ir*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    h(nx,ny)=x(i,j)
    h(nx+1,ny)=x(i,j)
    h(nx-1,ny)=x(i,j)
    h(nx,ny+1)=x(i,j)
    h(nx,ny-1)=x(i,j)
    do 50 ik=2,irz
        id=ik
        m=ik
        n=0
        h(nx+m,ny)=x(i,j)
        h(nx-m,ny)=x(i,j)
        h(nx,ny+m)=x(i,j)
        h(nx,ny-m)=x(i,j)
        if(m .eq. 0) go to 50
        n=n+1
        mn=m**2+n**2
        nr2=id**2
        nr1=(id-1)**2
        if(mn .gt. nr1) then
            if(mn .le. nr2) then
                h(nx+m,ny+n)=x(i,j)
                h(nx-m,ny+n)=x(i,j)
                h(nx+m,ny-n)=x(i,j)

```

```

60
70

```

```

        h(nx-m,ny-n)=x(i,j)
        go to 70
        else
            m=m-1
            n=n-2
        go to 60
    endif
    else
        go to 70
    endif
50    continue
40    continue
return
end

```

```

subroutine fitto(i,ir,rz,ne,thetar)

```

```

c      It gives an average value for every pixel in an element
c      of odd-number diameter.

```

```

integer*2 h(512,512)
integer*2 ximg(150,250), xedge(150,250), x(150,250)
integer*2 ir

```

```

common /b1/ h
common /b2/ ximg, xedge, x

```

```

thetae=2.*thetar
irz=int(rz)+1
do 4 j=1,ne
    pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir*cos(pt)
    yp=256-ir*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    h(nx,ny)=x(i,j)
    do 5 ik=2,irz
        d=ik-0.5
        m=ik-1
        n=0
        h(nx+m,ny)=x(i,j)
        h(nx-m,ny)=x(i,j)
        h(nx,ny+m)=x(i,j)
        h(nx,ny-m)=x(i,j)
6        if(m .eq. 0) go to 5
7        n=n+1
        ds=m**2.+n**2.
        r2=d**2.
        r1=(d-1.)**2.
    enddo
enddo

```

```

        if(ds .ge. r1) then
            if(ds .lt. r2) then
                h(nx+m,ny+n)=x(i,j)
                h(nx-m,ny+n)=x(i,j)
                h(nx+m,ny-n)=x(i,j)
                h(nx-m,ny-n)=x(i,j)
                go to 7
            else
                m=m-1
                n=n-2
                go to 6
            endif
        else
            go to 7
        endif
    endif
5      continue
4      continue
      return
      end

```

```

subroutine planepar(plane)

```

c It prints the message.

```

integer plane

write(6,10)
10  format(/,'Select the display plane')
write(6,20)
20  format(1x,'Type " 0 " for Image plane')
write(6,30)
30  format(1x,'Type " 1 " for Computational plane')
read(5,*) plane
return
end

```

```

subroutine imgplane(ir,nr,ne,thetar)

```

c It displays an image in image plane.

```

integer*2 h(512,512)
integer*2 ximg(150,250), xedge(150,250), x(150,250)
integer*2 ir(150)

common /b1/ h

```

```

common /b2/ ximg, xedge, x

thetae=2.*thetar
do 30 i=1,nr
  do 40 j=1,ne
    pt=(j-1)*thetac+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir(i)*cos(pt)
    yp=256-ir(i)*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    h(nx,ny)=x(i,j)
    h(nx+1,ny)=x(i,j)
    h(nx-1,ny)=x(i,j)
    h(nx,ny+1)=x(i,j)
    h(nx,ny-1)=x(i,j)
40    continue
30  continue
return
end

```

```

subroutine compuplane(nr,ne)

```

c It displays an image in computation plane.

```

integer*2 h(512,512)
integer*2 ximg(150,250), xedge(150,250), x(150,250)
integer   xi, yi, x1, x2, y1, y2

common /b1/ h
common /b2/ ximg, xedge, x

do 2 i=1,512
  do 1 j=1,512
    h(i,j)=0
1    continue
2  continue
i=1
if(ne .gt. 84) then
  lr=1
  ls=2
else
  lr=3
  ls=5
endif
ld=-(lr*2)
y1=lr+1
y2=nr*ls
x2=512+ne*ld
do 32 yi=y1,y2,ls

```

```

x1=512-(((1-(-1)**i)/2*lr + (1-(-1)**(i+1))/2*(-ld))
j=1
do 31 xi=x1,x2,ld
  h(xi,yi)=x(i,j)
  h(xi+1,yi)=x(i,j)
  h(xi-1,yi)=x(i,j)
  h(xi,yi+1)=x(i,j)
  h(xi,yi-1)=x(i,j)
  do 36 ik=2,lr
    id=ik
    m=ik
    n=0
    h(xi+m,yi)=x(i,j)
    h(xi-m,yi)=x(i,j)
    h(xi,yi+m)=x(i,j)
    h(xi,yi-m)=x(i,j)
34    if(m .eq. 0) go to 36
35    n=n+1
    mn=m**2+n**2
    nr2=id**2
    nrl=(id-1)**2
    if(mn .gt. nrl) then
      if(mn .le. nr2) then
        h(xi+m,yi+n)=x(i,j)
        h(xi-m,yi+n)=x(i,j)
        h(xi+m,yi-n)=x(i,j)
        h(xi-m,yi-n)=x(i,j)
        go to 35
      else
        m=m-1
        n=n-2
        go to 34
      endif
    else
      go to 35
    endif
36    continue
    j=j+1
31    continue
    i=i+1
32    continue
  return
end

```

subroutine fovea(ir1,color)

c It displays a fovea in an image.

integer*2 h(512,512)

```

integer*2 ir1
integer color

common /b1/ h

nx=256
ny=256
irz=ir1+1
h(nx,ny)=color
do 50 ik=2,irz
  id=ik
  m=ik-1
  n=0
  h(nx+m,ny)=color
  h(nx-m,ny)=color
  h(nx,ny+m)=color
  h(nx,ny-m)=color
60  if(m .eq. 0) go to 50
70  n=n+1
  mn=m**2+n**2
  nr2=id**2
  nr1=(id-1)**2
  if(mn .gt. nr1) then
    if(mn .le. nr2) then
      h(nx+m,ny+n)=color
      h(nx-m,ny+n)=color
      h(nx+m,ny-n)=color
      h(nx-m,ny-n)=color
    go to 70
    else
      m=m-1
      n=n-2
      go to 60
    endif
  else
    go to 70
  endif
50  continue
  return
end

```

```

subroutine opdisp(x0,y0,Rows,Cols)

```

```

c      Set up the displaying parameter for Lexidata.

```

```

integer Err(2),x0,y0,Rows,Cols

```

```

call dsopn(3,Err)
call dspld(-1)

```



```

call dscfg(639,479,10)
call dschan(255,255,255)
call dscsl(10,0,0)
call dscer
call dsllu(0,0,4095,0)
call dsllu(1024,0,1279,255)
call dsllu(2048,0,2303,255)
call dsllu(3072,0,3327,255)
call dslim(x0,y0,x0+Cols-1,y0+Rows-1)
return
end

```

```

subroutine wrim(nrow,ncol)

```

c Write an image to a file.

```

integer*2 h(512,512)
integer*2 col(512)
integer fp

common /b1/ h

call wrhead(nrow,ncol,fp)
do 20 j=1,ncol
  do 10 i=1,nrow
    col(i)=h(i,j)
10    continue
  call wrcol(col,nrow,fp)
20    continue
  return
end

```

```

subroutine thresh(nr,ne,level)

```

c It thresholds an image.

```

integer*2 ximg(150,250),xedge(150,250),x(150,250)
integer level

common /b2/ ximg,xedge,x

do 20 i=1,nr
  do 10 j=1,ne
    if (x(i,j) .ge. level) then
      x(i,j)=255

```

```

        else
            x(i,j)=0
        endif
10      continue
20      continue
        return
    end

```

```

subroutine zerocross(nr,ne)

```

```

c      Finding zerocrossing-points.

```

```

integer*2 ximg(150,250), xedge(150,250), x(150,250)
logical a1,a2,a3,a4,a5,a6,a7

```

```

common /b2/ ximg, xedge, x

```

```

do 20 i=1,nr
    k=(i+1)/2-i/2
    do 10 j=1,ne
        if(xedge(i,j) .gt. 0) then
            if(k .eq. 0)then
                a1=xedge(i,j-1) .lt. 0
                a2=xedge(i,j+1) .lt. 0
                a3=xedge(i+1,j) .lt. 0
                a4=xedge(i+1,j+1) .lt. 0
                a5=xedge(i-1,j) .lt. 0
                a6=xedge(i-1,j+1) .lt. 0
                a7=a1 .or. a2 .or. a3 .or. a4 .or. a5 .or. a6
                if(a7 .eq. .true.) then
                    x(i,j)=xedge(i,j)
                else
                    x(i,j)=0
                endif
            else
                a1=xedge(i,j-1) .lt. 0
                a2=xedge(i,j+1) .lt. 0
                a3=xedge(i+1,j-1) .lt. 0
                a4=xedge(i+1,j) .lt. 0
                a5=xedge(i-1,j-1) .lt. 0
                a6=xedge(i-1,j) .lt. 0
                a7=a1 .or. a2 .or. a3 .or. a4 .or. a5 .or. a6
                if(a7 .eq. .true.) then
                    x(i,j)=xedge(i,j)
                else
                    x(i,j)=0
                endif
            endif
        endif
    else

```

```

        x(i,j)=0
      endif
10      continue
20      continue
      return
      end

```

```

      subroutine rdimg(nr,ne)

```

```

c      It reads image data which is generated by the program
c      of human PVS.

```

```

      integer*2 ximg(150,250),xedge(150,250),x(150,250)

      common /b2/ ximg,xedge,x

      do 20 i=1,nr
        do 10 j=1,ne
          read(1,*) ximg(i,j)
          x(i,j)=ximg(i,j)
10        continue
20      continue
      return
      end

```

```

      subroutine rdedge(nr,ne)

```

```

c      It reads edge data which is generated by the program
c      of human PVS.

```

```

      integer*2 ximg(150,250),xedge(150,250),x(150,250)

      common /b2/ ximg,xedge,x

      do 20 i=1,nr
        do 10 j=1,ne
          read(2,*) xedge(i,j)
10        continue
20      continue
      return
      end

```

program nedge

c This program simulates three edge detection
c schemes for the new sensor.

```
integer*2 f(-20:120,-20:170),g(-20:120,-20:170),h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer*2 lm(300), ln(100), histo(0:255)
integer Rows,Cols,level,plane,rz1
integer t1,t2,t3,d
character*40 filename
logical lexon
```

```
common lm
common /b1/ f, g, h
common /b2/ op, edge
common /b3/ ln
```

```
lexon=.false.
Rows=512
Cols=512
maxgry=255
rz1=1
ir1=18
of=0.2
thetar=0.04425
nr=34
ne=71
call num
do 2 i=-20,120
  do 1 j=-20,170
    f(i,j)=0
```

```
1 continue
2 continue
```

```
5 write(6,10)
write(6,20)
write(6,30)
write(6,40)
write(6,50)
write(6,60)
write(6,70)
write(6,80)
write(6,90)
write(6,100)
write(6,110)
write(6,120)
write(6,130)
write(6,140)
write(6,150)
write(6,160)
write(6,170)
```

```
10 format(//,' Your options are: ')
```

```

20  format(/,'      1=Create an image')
30  format('      2=Input an image from camera')
40  format('      3=Input the parameters')
50  format('      4=Display the original image')
60  format('      5=Display the circular fitting image')
70  format('      6=Absolute value edge operation')
80  format('      7=Laplacian edge operation')
90  format('      8=Lap-Gaussian edge operation')
100 format('      9=display the operated image')
110 format('     10=display the edge without thresholding')
120 format('     11=display the edge with thresholding')
130 format('     12=write the image to a file')
140 format('     13=Read an image from Jays file')
150 format('     14=Write the Jays image to a file')
160 format('     15=quit')
170 format(/,' Selection?  ', $)
    read(5,*) imenu

    go to(200,250,300,350,400,500,600,700,800,900,1000,1100,
+      1200,1300,9999) imenu
    go to 5
200  call square
    go to 5
250  call inputim
    go to 5
300  call paramet(ir1,rz1,nr,ne,of,thetar)
    go to 5
350  call display(Rows,Cols,lexon)
    go to 5
400  call operation(ir1,rz1,nr,ne,of,thetar)
    call planeplane(plane)
    if(plane .eq. 0) then
        call imgplane(ir1,rz1,nr,ne,of,thetar)
    else
        call compuplane(nr,ne)
    endif
    call display(Rows,Cols,lexon)
    go to 5
500  call absop(nr,ne,histo)
    go to 5
600  call lapop(nr,ne)
    call zerocross(nr,ne,histo)
    go to 5
700  write(6,710)
710  format(/,'Please input the central region radius of weighting
+    function, rz')
    read(5,*) rz
    write(6,720) rz
720  format(1x,'rz= ',f7.1)
    call gusop(nr,ne,rz)
    call zerocross(nr,ne,histo)
    go to 5
800  do 820 i=1,nr

```

```

do 810 j=1,ne
  if(op(i,j) .gt. 0) then
    g(i,j)=255
  else
    if(op(i,j) .lt. 0) then
      g(i,j)=0
    else
      g(i,j)=127
    endif
  endif
810   continue
820   continue
  call planepar(plane)
  if(plane .eq. 0) then
    call imgplane(ir1,rz1,nr,ne,of,thetar)
  else
    call compuplane(nr,ne)
  endif
  call display(Rows,Cols,lexon)
  go to 5
900   do 920 i=1,nr
      do 910 j=1,ne
        g(i,j)=edge(i,j)
910     continue
920     continue
  call planepar(plane)
  if(plane .eq. 0) then
    call imgplane(ir1,rz1,nr,ne,of,thetar)
  else
    call compuplane(nr,ne)
  endif
  call display(Rows,Cols,lexon)
  go to 5
1000  write(6,1010)
      write(6,1020)
      write(6,1030)
      write(6,1040)
      write(6,1050)
      write(6,1060)
1010  format(//,' Thresholding Option')
1020  format('      1=Display current histogram')
1030  format('      2=Interactive thresholding')
1040  format('      3=d-peak thresholding')
1050  format('      4=Return to main menu')
1060  format(/,' Selection? ',%)
      read(5,*) imenu
      goto(5100,5200,5300,5) imenu
      go to 1000

5100  call disphs(histo,maxgry,100,450,150)
      go to 1000
5200  write(6,5210)
5210  format(//,' Select a Threshold Level: ',%)

```

```

read(5,*) level
call thresh(nr,ne,level)
call planepar(plane)
if(plane .eq. 0) then
  call imgplane(ir1,rz1,nr,ne,of,thetar)
else
  call compuplane(nr,ne)
endif
call display(Rows,Cols,lexon)
go to 1000
5300 write(6,5310) maxgry
5310 format(//,' Input a d value between 2 and ',i4,' : ', $)
read(5,*) d
if ((d .lt. 2) .or. (d .gt. maxgry)) go to 5300
call dpeak(histo,t1,t2,t3,maxgry,d)

if (t1 .eq. -1) then
  write(6,5320)
else
  write(6,5330) t1
  if (t2 .eq. -1) then
    write(6,5360)
  else
    write(6,5340) t2
    if (t3 .eq. -1) then
      write(6,5360)
    else
      write(6,5350) t3
    endif
  endif
endif
write(6,5210)
read(5,*) level
call thresh(nr,ne,level)
call planepar(plane)
if(plane .eq. 0) then
  call imgplane(ir1,rz1,nr,ne,of,thetar)
else
  call compuplane(nr,ne)
endif
call display(Rows,Cols,lexon)
endif
go to 1000
5320 format(//,' There are no thresholds for the specified d.')
5330 format(//,' First threshold at: ',i4)
5340 format(//,' Second threshold at: ',i4)
5350 format(//,' Third threshold at: ',i4)
5360 format(//,' There are no more thresholds for the specified d.')

1100 call wrimg(Rows,Cols)
go to 5
1200 write(6,1210)
1210 format(//,'What is the name of file: '$)
read(5,*)filename

```

```

close(1,err=1220)
1220 open(1,file=filename,err=1200)
call rdjay(nr,ne)
go to 5
1300 write(6,1310)
1310 format(//,'What is the filename: ', $)
read(5,*)filename
close(2,err=1320)
1320 open(2,file=filename,err=1300)
call writej(nr,ne)
go to 5
9999 stop
end

```

```

subroutine absop(nr,ne,histo)

```

c It performs the absolute value method for edge detection.

```

integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer*2 histo(0:255)
integer a1,a2,a3,a4,a5,a6
integer x, y

common /b1/ f, g, h
common /b2/ op, edge

nring=nr
nelem=ne
do 100 i=1,256
    histo(i-1)=0
100 continue
do 3 i=1,nring
    op(i,1)=0
    op(i,nelem)=0
    edge(i,1)=0
    edge(i,nelem)=0
3 continue
do 4 j=1,nelem
    op(1,j)=0
    op(nring,j)=0
    edge(1,j)=0
    edge(nring,j)=0
4 continue
x=nring-1
y=nelem-1
do 5 i=2,x
    do 6 j=2,y
        k=(i+1)/2-i/2

```



```

        if(k .eq. 0) then
            a1=f(i,j)-f(i+1,j)
            a2=f(i,j)-f(i,j-1)
            a3=f(i,j)-f(i-1,j)
            a4=f(i,j)-f(i-1,j+1)
            a5=f(i,j)-f(i,j+1)
            a6=f(i,j)-f(i+1,j+1)
            op(i,j)=abs(a1)+abs(a2)+abs(a3)+abs(a4)+abs(a5)+abs(a6)
            edge(i,j)=op(i,j)
        else
            a1=f(i,j)-f(i+1,j-1)
            a2=f(i,j)-f(i,j-1)
            a3=f(i,j)-f(i-1,j-1)
            a4=f(i,j)-f(i-1,j)
            a5=f(i,j)-f(i,j+1)
            a6=f(i,j)-f(i+1,j)
            op(i,j)=abs(a1)+abs(a2)+abs(a3)+abs(a4)+abs(a5)+abs(a6)
            edge(i,j)=op(i,j)
        endif
        histo(edge(i,j))=histo(edge(i,j))+1
6      continue
5    continue
      return
      end

```

subroutine square

c Create a square image.

```

integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)

common /b1/ f, g, h

do 1 i=1,512
  do 2 j=1,512
    if(i .ge. 150 .and. i .le. 350) then
      if(j .ge. 150 .and. j .le. 350) then
        h(i,j)=200
      else
        h(i,j)=125
      endif
    else
      h(i,j)=125
    endif
  2   continue
1    continue
      return
      end

```

```
subroutine compuplane(nr,ne)
```

c It displays an image in computation plane.

```
integer*2 lm(300)
integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer xi, yi, x1, x2, y1, y2

common lm
common /b1/ f, g, h
common /b2/ op, edge

do 2 i=1,512
  do 1 j=1,512
    h(i,j)=0
1    continue
2  continue
  i=1
  if(ne .gt. 84) then
    lr=1
    ls=2
  else
    lr=3
    ls=5
  endif
  ld=-(lr*2)
  y1=lr+1
  y2=nr*ls
  x2=512+ne*ld
  do 32 yi=y1,y2,ls
    x1=512-((1-(-1)**i)/2*lr + (1-(-1)**(i+1))/2*(-ld))
    j=1
    do 31 xi=x1,x2,ld
      do 36 ik=2,y1
        id=ik-1
        m=ik-1
        n=0
        if(m .eq. 1 .and. n .eq. 0) then
          h(xi,yi)=g(i,j)
          h(xi+m,yi)=g(i,j)
          h(xi-m,yi)=g(i,j)
        else
          h(xi+m,yi)=g(i,j)
          h(xi-m,yi)=g(i,j)
        endif
34      if(m .eq. 0) go to 36
        m=m-1
35      n=n+1
        mn=m**2+n**2
        nr2=id**2
```

```

        nrl=(id-1)**2
        if(mn .gt. nrl) then
if(mn .le. nr2) then
    if(m .eq. 0 .and. n .ne. 0) then
        h(xi,yi+n)=g(i,j)
        h(xi,yi-n)=g(i,j)
        go to 36
    else
        h(xi+m,yi+n)=g(i,j)
        h(xi-m,yi+n)=g(i,j)
        h(xi+m,yi-n)=g(i,j)
        h(xi-m,yi-n)=g(i,j)
        go to 35
    endif
else
    n=n-2
    go to 34
endif
        else
            go to 35
        endif
36      continue
        j=j+1
31      continue
        i=i+1
32      continue
        return
        end

```

```

subroutine gusop(nr,ne,rz)

```

```

c      It performs the Laplacian-Gaussian function method for
c      the edge detection.

```

```

integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer*2 ln(100)
real      w(50)

```

```

common /b1/ f, g, h
common /b2/ op, edge
common /b3/ ln

```

```

call numco
do 800 i=1,nr
    do 700 j=1,ne
        op(i,j)=0
        edge(i,j)=0
700    continue

```

```

800  continue
      irz=int(4.*rz)+1
      w(1)=0.
      do 10 i=1,irz
        rd=i-0.5
        w(i+1)=(rd**2.)*exp(-rd**2./rz**2.)
10    continue
      im=irz-1
      do 20 i=1,im
        w(i)=w(i+1)-w(i)
20    continue
      pi=3.14159
      nring=nr
      nelem=ne
      do 40 nr=1,nring,2
        do 30 ne=1,nelem
          t=f(nr,ne)*w(1)
          j=2
          sl=sin(pi/3.)
          do 1 m=1,im,1
            s=0.
            jd=1
            r=m+0.5
            k=m+1
            s=s+f(nr,ne+m)+f(nr,ne-m)
3          tl=jd*sl
            if(tl**2. .ge. r**2.) go to 9
            n=(jd+1)/2-jd/2
            if(n .eq. 0) then
              j1=0
            else
              j1=1
            endif
            do 2 i=j,j1,-1
              if(n .eq. 0) then
                i2=i
              else
                i2=i-1
              endif
              dj=(1-(-1)**jd)/2*0.5+float(i2)
              d=(tl**2.+dj**2.)*0.5
              d1=r-1.
              if(d .ge. d1) then
                if(d .lt. r) then
                  if(n .ne. 0) then
                    s=s+f(nr+jd,ne-i)+f(nr+jd,ne+(i-1))+f(nr-jd,ne-i)+
+                     f(nr-jd,ne+(i-1))
                  else
                    if(i .eq. 0) then
                      s=s+f(nr+jd,ne)+f(nr-jd,ne)
                    else
                      s=s+f(nr+jd,ne+i)+f(nr+jd,ne-i)+f(nr-jd,ne+i)+
+                     f(nr-jd,ne-i)

```

```

endif
endif
endif
endif
2      continue
      jd=jd+1
      go to 3
9      j=j+1
      t=t+s*w(k)/ln(k)
1      continue
      if(t .gt. -0.01 .and. t .lt. 0.01) then
        op(nr,ne)=0
      else
        if(t .gt. 0) then
          op(nr,ne)=int(t+1)
        else
          op(nr,ne)=int(t-1)
        endif
      endif
30     continue
40     continue
      iri=irz+1
      do 400 nr=2,nring,2
        do 300 ne=1,nelem
          t=f(nr,ne)*w(1)
          j=1
          sl=sin(pi/3.)
          do 11 m=1,im,1
            s=0.
            jd=1
            r=m+0.5
            k=m+1
            s=s+f(nr,ne+m)+f(nr,ne-m)
33      tl=jd*sl
            if(tl**2. .ge. r**2.) go to 99
            do 22 i=j,0,-1
              n=(jd+1)/2-jd/2
              dj=(1-(-1)**jd)/2*0.5+float(i)
              d=(tl**2.+dj**2. )**0.5
              d1=r-1.
              if(d .ge. d1) then
                if(d .lt. r) then
                  if(n .ne. 0) then
                    s=s+f(nr+jd,ne-i)+f(nr+jd,ne+(i+1))+f(nr-jd,ne-i)+
+                      f(nr-jd,ne+(i+1))
                  else
                    if(i .eq. 0) then
                      s=s+f(nr+jd,ne)+f(nr-jd,ne)
                    else
                      s=s+f(nr+jd,ne+i)+f(nr+jd,ne-i)+f(nr-jd,ne+i)+
+                      f(nr-jd,ne-i)
                    endif
                  endif
                endif
            enddo
          enddo
        enddo
      enddo

```

```

                endif
            endif
22          continue
            jd=jd+1
            go to 33
99          j=j+1
            t=t+s*w(k)/ln(k)
11         continue
            if(t .gt. -0.01 .and. t .lt. 0.01) then
                op(nr,ne)=0
            else
                if(t .gt. 0) then
                    op(nr,ne)=int(t+1)
                else
                    op(nr,ne)=int(t-1)
                endif
            endif
        endif
300        continue
400        continue
end

```

```

subroutine imgplane(ir1,rz1,nr,ne,of,thetar)

```

c It displays an image in image plane.

```

integer*2 lm(300)
integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer   rz,rz1

common lm
common /b1/ f, g, h
common /b2/ op, edge

r=float(ir1)
ir=ir1
rz=rz1
irz=rz+1
thetae=2.*thetar
a=(tan(thetar))**2.+1./cos(thetar)
do 30 i=1,nr
    do 40 j=1,ne
        pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
        xp=256+ir*cos(pt)
        yp=256-ir*sin(pt)
        nx=int(xp+0.5)
        ny=int(yp+0.5)
        do 50 ik=2,irz
            nt=nt+lm(ik-1)
        enddo
    enddo
enddo

```

```

        id=ik-1
        m=ik-1
        n=0
        if(m .eq. 1 .and. n .eq. 0) then
            h(nx,ny)=g(i,j)
            h(nx+m,ny)=g(i,j)
            h(nx-m,ny)=g(i,j)
        else
            h(nx+m,ny)=g(i,j)
            h(nx-m,ny)=g(i,j)
        endif
60      if(m .eq. 0) go to 50
        m=m-1
70      n=n+1
        mn=m**2+n**2
        nr2=id**2
        nr1=(id-1)**2
        if(mn .gt. nr1) then
            if(mn .le. nr2) then
                if(m .eq. 0 .and. n .ne. 0) then
                    h(nx,ny+n)=g(i,j)
                    h(nx,ny-n)=g(i,j)
                    go to 50
                else
                    h(nx+m,ny+n)=g(i,j)
                    h(nx-m,ny+n)=g(i,j)
                    h(nx+m,ny-n)=g(i,j)
                    h(nx-m,ny-n)=g(i,j)
                    go to 70
                endif
            else
                n=n-2
                go to 60
            endif
        else
            go to 70
        endif
50      continue
40      continue
        r=r*(a+(a**2.-1.)**0.5)
        ir=int(r+0.5)
        xrz=(r*sin(thetar))/(1.-of)
        rz=int(xrz+0.5)
        irz=rz+1
30      continue
        return
        end

```

subroutine lapop(nr,ne)

c It performs the Laplacian method for edge detection.

```
integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer a1,a2,a3,a4,a5,a6
integer x, y

common /b1/ f, g, h
common /b2/ op, edge

nring=nr
nelem=ne
do 3 i=1,nring
  op(i,1)=0
  op(i,nelem)=0
  edge(i,1)=0
  edge(i,nelem)=0
3 continue
do 4 j=1,nelem
  op(1,j)=0
  op(nring,j)=0
  edge(1,j)=0
  edge(nring,j)=0
4 continue
x=nring-1
y=nelem-1
do 5 i=2,x
  do 6 j=2,y
    k=(i+1)/2-i/2
    if(k .eq. 0) then
      a1=f(i,j)-f(i+1,j)
      a2=f(i,j)-f(i,j-1)
      a3=f(i,j)-f(i-1,j)
      a4=f(i,j)-f(i-1,j+1)
      a5=f(i,j)-f(i,j+1)
      a6=f(i,j)-f(i+1,j+1)
      op(i,j)=a1+a2+a3+a4+a5+a6
    else
      a1=f(i,j)-f(i+1,j-1)
      a2=f(i,j)-f(i,j-1)
      a3=f(i,j)-f(i-1,j-1)
      a4=f(i,j)-f(i-1,j)
      a5=f(i,j)-f(i,j+1)
      a6=f(i,j)-f(i+1,j)
      op(i,j)=a1+a2+a3+a4+a5+a6
    endif
  endif
6 continue
5 continue
return
end
```

subroutine num

c It gives the total number of pixels for an element.

integer*2 lm(300)

common lm

lm(1)=5

write(6,2) lm(1)

2 format(1x,'lm(1)= ',i3)

do 5 ik=3,151

l=ik-1

lm(1)=0

id=ik-1

m=ik-1

n=0

if(m .ne. 0 .and. n .eq. 0) then

lm(1)=lm(1)+1

endif

6 if(m .eq. 0) go to 8

m=m-1

7 n=n+1

mn=m**2+n**2

nr2=id**2

nr1=(id-1)**2

if(mn .gt. nr1) then

if(mn .le. nr2) then

if(m .eq. 0 .and. n .ne. 0) then

lm(1)=lm(1)+1

go to 8

else

lm(1)=lm(1)+1

go to 7

endif

else

n=n-2

go to 6

endif

else

go to 7

endif

8 lm(1)=4*(lm(1)-1)

write(6,1) 1, lm(1)

1 format(1x,'lm(',i3,')= ',i5)

5 continue

end

subroutine numco

c It gives the total number of elements in
c Laplacian-Gaussian function method.

```

integer*2 ln(100)

common /b3/ ln

pi=3.14159
j=1
sl=sin(pi/3.)
ln(1)=1
write(6,4)ln(1)
4 format(1x,'ln(1)= ',i3)
do 1 m=1,20,1
    jd=1
    r=m+0.5
    k=m+1
    ln(k)=0
    ln(k)=ln(k)+2
3    do 2 i=j,0,-1
        tl=jd*sl
        n=(jd+1)/2-jd/2
        if(tl**2. .ge. r**2.) go to 9
        dj=(1-(-1)**jd)/2*0.5+float(i)
        d=(tl**2.+dj**2.)*0.5
        d1=r-1.
        if(d .ge. d1)then
            if(d .lt. r)then
                if(i .eq. 0 .and. n .eq. 0) then
                    ln(k)=ln(k)+2
                else
                    ln(k)=ln(k)+4
                endif
            endif
        endif
        continue
2    jd=jd+1
    go to 3
9    j=j+1
    write(6,5)k,ln(k)
5    format(1x,'ln(',i2,')= ',i3)
1    continue
end

```

subroutine operation(ir1,rz1,nr,ne,of,thetar)

c It gives an average value for every element in the new sensor.

```

integer*2 lm(300)
integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer    rz,rz1

```

```

common lm
common /b1/ f, g, h
common /b2/ op, edge

```

```

r=float(ir1)
ir=ir1
rz=rz1
irz=rz+1
thetae=2.*thetar
a=(tan(thetar))*2.+1./cos(thetar)
do 3 i=1,nr
  do 4 j=1,ne
    pt=(j-1)*thetae+(1.-(-1)**(i+1))/2.*thetar
    xp=256+ir*cos(pt)
    yp=256-ir*sin(pt)
    nx=int(xp+0.5)
    ny=int(yp+0.5)
    ft=0.
    nt=0
    do 5 ik=2,irz
      s=0.
      nt=nt+lm(ik-1)
      id=ik-1
      m=ik-1
      n=0
      if(m .eq. 1 .and. n .eq. 0) then
        s=s+h(nx+m,ny)+h(nx-m,ny)+h(nx,ny)
      else
        s=s+h(nx+m,ny)+h(nx-m,ny)
      endif
      if(m .eq. 0) go to 8
      m=m-1
      n=n+1
      mn=m**2+n**2
      nr2=id**2
      nr1=(id-1)**2
      if(mn .gt. nr1) then
        if(mn .le. nr2) then
          if(m .eq. 0 .and. n .ne. 0) then
            s=s+h(nx,ny+n)+h(nx,ny-n)
            go to 8
          else
            s=s+h(nx+m,ny+n)+h(nx-m,ny+n)+h(nx+m,ny-n)
            +h(nx-m,ny-n)
            go to 7
          endif
        else
          +
          go to 8
        endif
      else
        +
        go to 8
      endif
    enddo
  enddo
enddo

```

```

        n=n-2
        go to 6
    endif
    else
        go to 7
    endif
8      ft=ft+s
5      continue
        fav=ft/nt
        f(i,j)=int(fav+0.5)
        g(i,j)=f(i,j)
4      continue
        r=r*(a+(a**2.-1.）**0.5)
        ir=int(r+0.5)
        xrz=(r*sin(thetar))/(1.-of)
        rz=int(xrz+0.5)
        irz=rz+1
3      continue
        return
    end

```

```

subroutine paramet(ir1,rz1,nr1,ne,of,thetar)

```

c It gives the number of elements per ring, the number
c of rings and their coordinates.

```

integer*2 rz(100), ir(100)
integer*2 rz1

write(6,1)
1  format(/,'Please input the smallest integer value of the central
+ region of the overlapping receptive field, rz(1)')
  read(5,*) rz(1)
  rz1=rz(1)
  write(6,2) rz(1)
2  format(1x,'rz(1)= ',i1)
  write(6,3)
3  format(/,'Please input the smallest integer value of the eccentric
+ ity, ir(1)')
  read(5,*) ir(1)
  ir1=ir(1)
  write(6,4) ir(1)
4  format(1x,'ir(1)= ',i2)
  write(6,5)
5  format(/,'Please input the overlapping factor, of')
  read(5,*) of
  write(6,6) of
6  format(1x,'of= ',f10.5)
  pi=3.14159

```

```

r=float(ir(1))
rzn=(1.-of)*rz(1)
rn=pi/asin(rzn/r)
ne=int(rn+0.5)
thetar=pi/ne
a=(tan(thetar))**2.+1./cos(thetar)
i=1
9  if((ir(i)+rz(i)) .le. 256) then
    if((ir(i)+4*rz(i)) .le. 256) then
        nr2=i
    endif
    r=r*(a+(a**2.-1.)**0.5)
    xrz=(r*sin(thetar))/(1.-of)
    i=i+1
    ir(i)=int(r+0.5)
    rz(i)=int(xrz+0.5)
    go to 9
endif
nr1=i-1
write(6,7) ne
7  format(/,'Number of elements per ring, ne= ', i3)
write(6,8) thetar
8  format(/,'thetar= ',f10.5)
write(6,11) nr1
11 format(/,'Number of rings for image, nr1= ',i2)
write(6,12) nr2
12 format(/,'Number of rings for edge, nr2= ',i2)
return
end
subroutine planepar(plane)
integer plane
write(6,10)
10 format(/,'Select the display plane')
write(6,20)
20 format(1x,'Type " 0 " for Image plane')
write(6,30)
30 format(1x,'Type " 1 " for Computational plane')
read(5,*) plane
return
end

```

```

subroutine rdjay(nr,ne)

```

c It reads the data from Jay's file.

```

integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 nr, ne

```

```

common /b1/ f, g, h

```

```

        read(1,*)ne,nr
        write(6,*)nr,ne
        do 20 i=1,nr
            do 10 j=1,ne
                read(1,*)f(i,j)
                g(i,j)=f(i,j)
10          continue
20        continue
        return
        end

```

```

subroutine opdisp(x0,y0,Rows,Cols)

```

c Setup the displaying parameters for Lexidata device.

```

integer Err(2),x0,y0,Rows,Cols
call dsopn(3,Err)
call dspld(-1)
call dscfg(639,479,10)
call dschan(255,255,255)
call dscsl(10,0,0)
call dscer
call dsllu(0,0,4095,0)
call dsllu(1024,0,1279,255)
call dsllu(2048,0,2303,255)
call dsllu(3072,0,3327,255)
call dslim(x0,y0,x0+Cols-1,y0+Rows-1)
return
end

```

```

subroutine thresh(nr,ne,level)

```

c It thresholds an image.

```

integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer level

common /b1/ f, g, h
common /b2/ op, edge

do 20 i=1,nr
    do 10 j=1,ne
        if (edge(i,j) .ge. level) then
            g(i,j)=255

```

```

        else
          g(i,j)=0
        endif
10      continue
20      continue
      return
    end

```

```

subroutine writej(nr,ne)

```

c Write the data to Jay's file for display.

```

integer*2 f(-20:120,-20:170),g(-20:120,-20:170), h(512,512)
integer    ne,nr

common /b1/ f, g, h

write(2,*)ne,nr
do 20 i=1,nr
  do 10 j=1,ne
    write(2,*)g(i,j)
10    continue
20    continue
  return
end

```

```

subroutine wrimg(nrow,ncol)

```

c It writes the image to a file.

```

integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 col(512)
integer fp

common /b1/ f, g, h

call wrhead(nrow,ncol,fp)
do 20 j=1,ncol
  do 10 i=1,nrow
    col(i)=h(i,j)
10    continue
  call wrcol(col,nrow,fp)
20    continue
  return
end

```

```
subroutine zerocross(nr,ne,histo)
```

c It finds the zerocrossing-points.

```
integer*2 op(-20:120,-20:170), edge(-20:120,-20:170)
integer*2 histo(0:255)
logical b1,b2,b3,b4,b5,b6,b7
```

```
common /b2/ op, edge
```

```
do 100 i=1,256
  histo(i-1)=0
100 continue
nring=nr
nelem=ne
do 20 i=1,nring
  k=(i+1)/2-i/2
  do 10 j=1,nelem
    if(op(i,j) .gt. 0) then
      if(k .eq. 0)then
        b1=op(i+1,j) .lt. 0
        b2=op(i,j-1) .lt. 0
        b3=op(i-1,j) .lt. 0
        b4=op(i-1,j+1) .lt. 0
        b5=op(i,j+1) .lt. 0
        b6=op(i+1,j+1) .lt. 0
        b7=b1 .or. b2 .or. b3 .or. b4 .or. b5 .or. b6
        if(b7 .ne. .true.) then
          edge(i,j)=0
        else
          edge(i,j)=op(i,j)
        endif
      else
        b1=op(i+1,j-1) .lt. 0
        b2=op(i,j-1) .lt. 0
        b3=op(i-1,j-1) .lt. 0
        b4=op(i-1,j) .lt. 0
        b5=op(i,j+1) .lt. 0
        b6=op(i+1,j) .lt. 0
        b7=b1 .or. b2 .or. b3 .or. b4 .or. b5 .or. b6
        if(b7 .ne. .true.) then
          edge(i,j)=0
        else
          edge(i,j)=op(i,j)
        endif
      endif
    else
      edge(i,j)=0
    endif
    histo(edge(i,j))=histo(edge(i,j))+1
10 continue
```



```
20      continue
      return
      end
```

subroutine inputim

c Input an image from a camera.

```
integer*2 f(-20:120,-20:170), g(-20:120,-20:170), h(512,512)
integer*2 col(512)
integer fp
```

```
common /b1/ f, g, h
```

```
call rdhead(nrow,ncol,fp)
do 1 j=1,512
    call rdcol(col,nrow,fp)
    do 2 i=1,512
        h(i,j)=col(i)
```

```
2      continue
1      continue
      return
      end
```

subroutine dispsh(histo,maxgry,xo,yo,maxy)

c Display the histogram of an image.

```
integer histo(0:255),maxgry,xo,yo,maxy
integer erase,color,length,gray
```

```
real      scale
```

```
10      write(6,20)
```

```
20      format(//,'Input a scale factor (-1 to quit); '$)
```

```
read(5,*) scale
```

```
if (scale .le. 0) go to 999
```

```
do 40 erase=1,2
```

```
    if (erase .eq. 1) then
```

```
        color=0
```

```
    else
```

```
        color=255
```

```
    endif
```

```
    call dsvec(xo,yo,xo+gray,yo,color)
```

```
    do 30 gray=0,maxgry
```

```
        if (erase .eq. 1) then
```

```
            length=maxy
```

```

        else
            length=histo(gray)/scale
            if (length .gt. maxy) length=maxy
        endif
        call dsvec(xo+gray,yo,xo+gray,yo-length,color)
30    continue
40    continue
999   return
      end

```

```

subroutine dpeak(histo,t1,t2,t3,maxgry,d)

```

c It finds the three suitable threshold values for an image.

```

real merit(128),minn
integer histo(0:255),t(3),peak(128),peaki(128)
integer valley(128),valleyi(128),d,maxgry,k,kmin
integer numpeaks,width,gray,winmax,jmax
integer t1,t2,t3

do 100 i=1,128
    peak(i)=0
    peaki(i)=0
    valley(i)=0
    valleyi(i)=0
    merit(i)=0
100  continue

k=0
width=2*d+1
do 120 i=0,maxgry
    winmax=0
    do 110 j=i-d,i+d
        if ((j .lt. 0) .or. (j .gt. 255)) then
            gray=0
        else
            gray=histo(j)
        endif
        if (gray .ge. winmax) then
            winmax=gray
            jmax=j
        endif
110    continue
        if (jmax .eq. i) then
            k=k+1
            peak(k)=winmax
            peaki(k)=i
        endif
120    continue

```

```

numpeaks=k
160 do 130 j=1,numpeaks-1
    valley(j)=100000
    valleyi(j)=0
    do 130 i=peaki(j)+1,peaki(j+1)-1
        if(histo(i) .le. valley(j)) then
            valley(j)=histo(i)
            valleyi(j)=i
        endif
130    continue

    do 140 j=1,numpeaks-1
        merit(j)=float(valley(j))/float(min(peak(j),peak(j+1)))
140    continue

    t(1)=-1
    t(2)=-1
    t(3)=-1
    if ((numpeaks-1) .lt. 3) then
        n=numpeaks-1
    else
        n=3
    endif
    do 145 j=1,n
        minn=1.1
        do 155 k=1,numpeaks-1
            if (merit(k) .lt. minn) then
                minn=merit(k)
                kmin=k
            endif
155        continue
        t(j)=valleyi(kmin)
        merit(kmin)=5.0
145    continue

    t1=t(1)
    t2=t(2)
    t3=t(3)
    return
end

```

```

subroutine display(Rows,Cols,lexon)

```

c Display an image.

```

integer*2 f(-20:120,-20:170),g(-20:120,-20:170),h(512,512)
integer Rows,Cols,x,y
integer*2 Row(512)

```

```

logical lexon

common /b1/ f,g,h

if (.not. lexon) then
  call opdisp(0,0,Rows,Cols)
  lexon=.true.
endif
do 1 x=1,Rows
  do 2 y=1,Cols
    if (h(x,y) .lt. 0) then
      Row(y)=0
    else
      if(h(x,y) .gt. 255) then
        Row(y)=255
      else
        Row(y)=h(x,y)
      endif
    endif
  2 continue
  call dsput(Row,Cols)
1 continue
return
end

```

APPENDIX A.3

LIST OF PROGRAMS USED IN CHAPTER III

```

Subroutine Zeroer(ncoun,zero,fixed,minpts)
c   Calculates the optimal zero level. That is finds that value
c   of zero which provides the fewest intersection points.
c
Integer ncoun, prevn, count, minpts
Logical fixed, teprev
Real zero, pzero(30)
c
pzero(0) = 0
count = count + 1
c
Print 160, count, ncoun, zero
c
If ((count .gt. 30) .or. (abs(zero-pzero(count-1)) .lt. 1e-8))
1   Go to 11
c
pzero(count) = zero
teprev = ((prevn .eq. ncoun) .and. (ncoun .ge. 3))
c
If ((ncoun .gt. 150) .or. teprev) Then
    fixed = .true.
    Goto 10
Endif
c
If (ncoun .gt. 3) Then
    zero = zero - abs((zero-pzero(count-1))/2.0)
    fixed = .false.
    Go to 10
Endif
c
If (ncoun .lt. 3) Then
    zero = zero + 1.1*abs((zero-pzero(count-1))/2.0)
    fixed = .false.
Else
    fixed = .true.
Endif
c
10   prevn = ncoun
c
    Go to 15
c
11   count = 0
    minpts = int(0.9*minpts)
c
15   Return
150  Format (/, 'Calculating optimal zero value...')
160  Format ('Pass ',i2,i5,g15.5)
End

```

```

Subroutine Writer(x,y,n,k)
c   Writes the data in a file. Strictly for error analysis.
c
c   Common nmax
c
c   Integer n, k
c   Real x(nmax), y(nmax)
c
c   If (k .gt. 4) Then
c       Print 110
c   Else
c       Do 10, i = 1, n
c           Write(k,100) i, x(i), i, y(i)
10      Continue
c   Endif
c   Return
100  Format ('x('i3, ') = 'g13.6,' y('i3,') = 'g13.6)
110  Format ('Cannot write data in files anymore')
End

```

```

Subroutine Trnslt(x,y,n)
c Translate the data set.
c
Common k, nmax
c
Real x(nmax), y(nmax)
c
xtrans = 0.0
ytrans = 0.0
c
Print 100
Read *, xtrans, ytrans
c
Do 10, i = 1,n
    x(i) = x(i) + xtrans
    y(i) = y(i) + ytrans
10 Continue
c
Write(k,110) xtrans, ytrans
c
Return
100 Format ('Enter the x,y translations --> ', $)
110 Format ('The translations are: ', 2f5.2)
End

```



```

Subroutine Sort(x,n,xmin,xmax)
c  Sorts the data set and returns the maximum and the minimum values.
c
c  Common nmax
c
c  Real x(nmax), tempx(1000)
c
c  Do 10, i = 1,n
    tempx(i) = x(i)
10  Continue
c
c  Do 120 nc = 2,n
    Do 100 i = nc,n
        If (tempx(i) .gt. tempx(nc-1)) Then
            temp = tempx(nc-1)
            tempx(nc-1) = tempx(i)
            tempx(i) = temp
        Endif
100  Continue
120  Continue
c
    xmax = tempx(1)
    xmin = tempx(n)
c
    Return
    End

```

Subroutine Setup(xmax,xmin,ymax,ymin,dev,a,b,csiz)

To setup screen parameters and draw the axes.

Real a,b,dev,csiz,xmax,xmin,ymax,ymin,axlenx,axleny

Real deltx,delty,xmnpr,ymnpr,ymxpr

xmnpr = xmin - 0.1*xmax

ymnpr = ymin - 0.1*ymax

ymxpr = 1.1*ymax

ymxpr = 1.1*ymax

Call plots(dev)

Call scrnp(a,b,a,-b)

Call paper(xmnpr,ymnpr,ymxpr)

axlenx = xmax - xmin

axleny = ymax - ymin

deltx = 0.2*axlenx

delty = 0.2*axleny

Call axis(xmin,ymin,1h,-1,axlenx,0,xmin,deltx,csiz,0,2,0,1)

Call axis(xmin,ymin,0h,0,axleny,90,ymin,delty,csiz,0,2,0,1)

Return

End

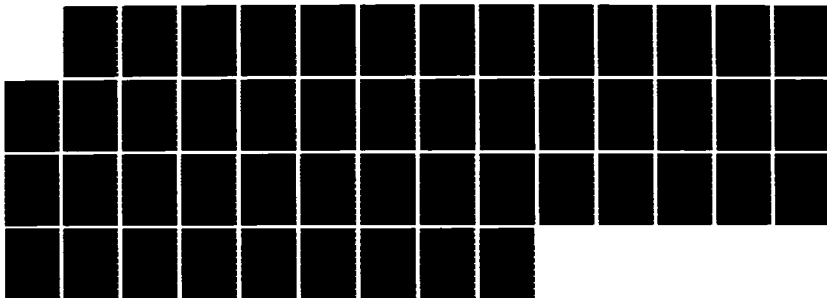
AD 16854

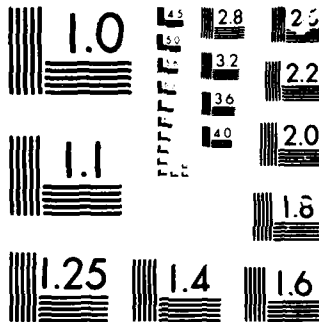
BIOLOGICAL VISUAL SYSTEMS STRUCTURES FOR MACHINE VISION 4/4
APPLIED TO ROBOT. (U) VIRGINIA UNIV CHARLOTTESVILLE
DEPT OF ELECTRICAL ENGINEERING. R M INIGO ET AL.

UNCLASSIFIED

FEB 86 UVA/525647/EE86/101 AFOSR-TR-86-0282 F/G 6/4

NL





```

Subroutine Secsrt(x,y,num)
c  Sorts data by one variable keeping the relative position constant.
c
Integer num, nc, i
Real x(1000), y(1000), tempx, tempy
c
Do 120 nc = 2,num
  Do 100 i = nc,num
    If (x(i) .gt. x(nc-1)) Then
      tempx = x(nc-1)
      x(nc-1) = x(i)
      x(i) = tempx
c
      tempy = y(nc-1)
      y(nc-1) = y(i)
      y(i) = tempy
c
    Endif
  100 Continue
120 Continue
  Return
End

```

```

Subroutine Rotate(x,y,n)
c   To rotate the data points
c
Common k, nmax
c
Real x(nmax), y(nmax), tempx, tempy, pi
c
pi = 3.1415926
c
Print 100
Read *, theta
c
If (theta .gt. 360) theta = theta - int(theta/360)*360
If (theta .lt. 0) theta = theta + int(theta/360)*360
c
Write (k,110) theta
c
theta = theta*pi/180.
c
Do 10, i = 1,n
    tempx = x(i)*cos(theta) - y(i)*sin(theta)
    tempy = y(i)*cos(theta) + x(i)*sin(theta)
    x(i) = tempx
    y(i) = tempy
10 Continue
c
Return
100 Format (/, 'Enter the angle of rotation --> ', $)
110 Format ('Angle of rotation = ', f6.2)
End

```

```

Subroutine Qualgc(tempx,tempy,num)
c   The qualitative algorithm to track direction of transformation
c   for images which the optical axis.
c   ** Still under construction **
c
Character*8 motion(2)
Integer num, nmax, k, cntxp, cntyp, cntxc, cntyc
Real tempx(6,nmax), tempy(6,nmax)
c
Common k, nmax
c
cntxp = 0
cntxc = 0
cntyp = 0
cntyc = 0
motion(1) = ''
motion(2) = ''
c
Do 10, i = 1, num
  If (tempx(k-1,i) .lt. tempx(k-2,i)) Then
    cntxp = cntxp + 1
  Else
    cntxc = cntxc + 1
  Endif
c
  If (tempy(k-1,i) .lt. tempy(k-2,i)) Then
    cntyp = cntyp + 1
  Else
    cntyc = cntyc + 1
  Endif
10 Continue
c
If (cntxp .gt. cntxc) Then
  motion(1) = 'left'
Else
  motion(1) = 'right'
Endif
c
If (cntyp .gt. cntyc) Then
  motion(2) = 'down'
Else
  motion(2) = 'up'
Endif
c
Do 20, i = 1, 2
  If (motion(i) .ne. '') Print 100, motion(i)
20 Continue
c
Return
100 Format ('Moved ', a8)
End

```

```

        cntcx = cntcx + 1
    Endif
c
50    Continue
c
    If (minim .eq. first) Goto 60
    first = minim
    second = maxim
    Goto 10
c
60    If ((cntc1-cntp2.gt.4).and.(cntp1-cntc2.gt.4)) motion(1)='right'
    If ((cntc2-cntp1.gt.4).and.(cntp2-cntc1.gt.4)) motion(2)='left'
    If ((cntc3-cntp4.gt.4).and.(cntp3-cntc4.gt.4)) motion(3)='up'
    If ((cntc4-cntp3.gt.4).and.(cntp4-cntc3.gt.4)) motion(4)='down'
c
    equal = ((cntc1 .eq. cntp2) .and. (cntp1 .eq. cntc2) .and.
1      (cntc3 .eq. cntp4) .and. (cntp3 .eq. cntc4))
c
    If (equal .and. (cntpx .lt. cntcx)) motion(5)='magnified right'
    If (equal .and. (cntpx .gt. cntcx)) motion(6)='reduced left'
c
    Do 70, i = 1, 6
        If (motion(i) .ne. "") Then
            Print 120, motion(i)
            Write(3,120) motion(i)
        Endif
70    Continue
c
    Return
120    Format ('Moved ', a32)
    End

```



```

Subroutine Qualg(stcurx,stcury,stprvx,stprvy,nmax,minim,maxim)
c The qualitative algorithm to track direction of transformation
c for images which do not contain the optical axis.
c ** Still under construction **
c
Character*32 motion(6)
Integer cntc1, cntc2, cntc3, cntc4, cntp1, cntp2, cntp3, cntp4
Integer cntcx, cntpx, minim, maxim, first, second
Logical equal
Real stcurx(nmax), stcury(nmax), stprvx(nmax), stprvy(nmax)
c
pi = 3.1415926
first = 1
second = minim
cntc1 = 0
cntc2 = 0
cntc3 = 0
cntc4 = 0
cntcx = 0
cntp1 = 0
cntp2 = 0
cntp3 = 0
cntp4 = 0
cntpx = 0
Do 5, i = 1, 6
    motion(i) = ""
5 Continue
c
10 Do 50, i = first, second
c
    If (((stcury(i) .ge. 3*pi/2.) .and. (stcury(i) .le. 2*pi))
1    .or. ((stcury(i) .ge. 0.) .and. (stcury(i) .lt. pi/2.)))
2    cntc1 = cntc1 + 1
    If ((stcury(i) .ge. pi/2.) .and. (stcury(i) .lt. 3*pi/2.))
1    cntc2 = cntc2 + 1
c
    If ((stprvy(i) .ge. pi/2.) .and. (stprvy(i) .lt. 3*pi/2.))
1    cntp1 = cntp1 + 1
    If (((stprvy(i) .ge. 3*pi/2.) .and. (stprvy(i) .le. 2*pi))
1    .or. ((stprvy(i) .ge. 0.) .and. (stprvy(i) .lt. pi/2.)))
2    cntp2 = cntp2 + 1
c
    If ((stcurx(i) .ge. 0.) .and. (stcurx(i) .lt. pi))
1    cntc3 = cntc3 + 1
    If ((stcurx(i) .ge. pi) .and. (stcurx(i) .lt. 2*pi))
1    cntc4 = cntc4 + 1
c
    If ((stprvy(i) .ge. pi) .and. (stprvy(i) .lt. 2*pi))
1    cntp3 = cntp3 + 1
    If ((stprvy(i) .ge. 0.) .and. (stprvy(i) .lt. pi))
1    cntp4 = cntp4 + 1
c
    If (stcurx(i) .lt. stprvx(i)) Then
        cntpx = cntpx + 1
    Else

```

```

Subroutine Menu(option,ans1)
c   Presents the menu
c
c   Character*1 ans, ans1
c   Integer option
c
c   option = 0
c
c   Print 100
c   Print 110
c   Print 120
c   Print 125
c   Read 140, ans
c
c   If ( ans .eq. '1') option = 1
c   If ( ans .eq. '2') option = 2
c   If ( ans .eq. '3') option = 3
c   If (option .ne. 0) Goto 10
c
c   If (ans .ne. 'n') Then
c       Print 130
c       Read 150, option
c   Else
c       option = 4
c   Endif
c
c   If (ans1 .eq. 'n') Call Clrscn
c
c   Return
100  Format (/, 'Would you like to do any of the following?')
110  Format ('  1. Magnify the figure')
120  Format ('  2. Rotate the figure ')
125  Format ('  3. Translate the figure: [y]/n ', $)
130  Format (/, 'Enter the option -> ', $)
140  Format (a1)
150  Format (i2)
End

```

```

95      Call Logmap(x,y,n,dev,a,b,csizexmax)
c
      Print 180
      Read 120, reply
c
      If (reply .eq. 'y') Goto 1
c
      If (reply .eq. 'c') Then
          n = ntem
          Do 20, i = 1, n
              x(i) = xtem(i)
              y(i) = ytem(i)
20      Continue
          Goto 95
      Endif
c
      If (reply .eq. 'x') Goto 5
c
      Else
c
          Print 140
c
          Endif
c
      Stop
100     Format (a80)
110     Format (' Do you want to see the mapped figure? ', $)
120     Format (a1)
140     Format ('Tough!!')
150     Format (i1)
160     Format ('/, 'Enter the screen parameters: ', $)
170     Format ('Enter the height of the numbering characters: ', $)
180     Format ('Do you want to conduct another session? ', $)
200     Format ('Do you want the output on:', $)
1       '      1) Tektronix 4014', $)
2       '      2) Plotter', $)
3       '      3) Tektronix 4107', $)
4       '      4) Laser Printer: ', $)
      End

```

Program Main

Main program. Calls Input1 and Logmap.

```
c
Common nmax
c
Character*80 header
Character*1 ans, reply
Integer n, ntem, dev
Real x(1000), y(1000), xtem(1000), ytem(1000)
Real a, b, csize, xmin, xmax, ymin, ymax
c
1 Call Clrscn
c
Print 200
Read 150, dev
c
If (dev .eq. 3) Then
    a = 0.20
    b = 0.80
    csize = .08
    Goto 4
Endif
c
Print 160
Read *, a, b
a = a/100.
b = b/100.
c
Print 170
Read *, csize
c
4 nmax = 1000
c
5 Call Input1(x,y,n,xmin,xmax,ymin,ymax,1,header)
ntem = n
Do 10, i = 1, n
    xtem(i) = x(i)
    ytem(i) = y(i)
10 Continue
c
Call Setup(xmax,xmin,ymax,ymin,dev,a,b,csize)
Call line(x,y,n,1,0,0,0)
Print 100, header
c
Call Waite
c
Call plot(0.,0.,999)
c
Print 110
Read 120, ans
c
Call Clrscn
c
If (ans .ne. 'n') Then
c
```

```

Subroutine Mgnfy(x,y,n)
c   To magnify the data set.
c
c   Common nmax
c
c   Real x(nmax), y(nmax), factor
c
c   Print 100
c   Read *, factor
c
c   Do 10, i = 1, n
c       x(i) = x(i)*factor
c       y(i) = y(i)*factor
10  Continue
c
c   Return
100 Format (/, 'Enter the magnification factor --> ', $)
End

```

```

64      Call Imdiff(tempx,tempy,num,xmax,xmin,
1          ymax,ymin,dev,a,b,csiz,centre,minchk)
c
73      Call Waite
c
      Endif
c
      Call plot(0.,0.,999)
c
      Return
100     Format (/, 'Would you like to do anything else? ', $)
110     Format (a1)
130     Format (/, 'Performing the transformation... ',/)
      End

```

```

      Endif
c
81  Do 10, i = 1, n
c
      If (abs(w(i)) .lt. 1.e-6) w(i) = 1.e-6
c
      x(i) = 0.5*alog(w(i)**2 + z(i)**2)
      y(i) = atan(z(i)/w(i))
c
      If (w(i) .lt. 0.) y(i) = y(i) + pi
c
      If (.not.(centre)) Then
        If (y(i) .lt. 0.) y(i) = y(i) + 2*pi
      Endif
c
10  Continue
c
      num = i - 2
c
16  Call Conver(tempx,x,num)
      Call Conver(tempy,y,num)
c
17  numim = numim + 1
c
      If (ans .ne. 'n') Goto 11
c
      Call Sort(x,num,xmin,xmax)
      Call Sort(y,num,ymin,ymax)
c
      maxy = ymax
      miny = 0.0
      minx = 0.0
      maxx = xmax
c
      Call Setup(maxx,minx,maxy,miny,dev,a,b,csiz)
c
11  xpt = x(1)
      ypt = y(1)
      minchk = (ypt .eq. ymin)
c
      If (.not.(centre .or. minchk)) Call Secsrt(y,x,num)
c
      Call line(x,y,num,1,0,0,k-1,0)
      Call number(xpt,ypt,.06,0.0,0,-1,0)
c
18  k = k + 1
c
      Call Waite
c
      Print 100
      Read 110, ans
      If (ans .ne. 'n') Goto 1
c
      If (numim .gt. 1) Then
c

```

```

Subroutine Logmap(w,z,n,dev,a,b,csiz,maxx)
c Subroutine to do the mapping
c
c Common k,nmax
c
c Character*1 ans
c Integer option,i,num,k,dev,numim
c Logical centre,cond1,cond2,cond3,cond4,minchk
c Real pi,a,b,csiz,zmin,zmax
c Real w(nmax),z(nmax),x(1000),y(1000),tempy(6,1000),tempx(6,1000)
c Real xmax,xmin,ymin,maxx,minx,maxy,miny,wmin,wmax
c
c pi = 3.1415926
c k = 1
c numim = 0
c num = 0
c ans = 'n'
c
c **** Initialise ****
c Do 25, j = 1, 6
c   Do 24, i = 1, n
c     tempx(j,i) = 0.0
c     tempy(j,i) = 0.0
24   Continue
25   Continue
c
c Call Clrscn
c
c 1 Call Menu(option,ans)
c   Goto (5,6,7,8) option
c
c 5 Call Mgnfy(w,z,n)
c   num = num + 1
c   Goto 8
c
c 6 Call Rotate(w,z,n)
c   Goto 8
c
c 7 Call Trnslt(w,z,n)
c
c 8 Print 130
c   If (ans .ne. 'n') Goto 81
c
c Call Sort(w,n,wmin,wmax)
c Call Sort(z,n,zmin,zmax)
c
c cond1 = (zmin .gt. 0.)
c cond2 = (zmax .lt. 0.)
c cond3 = (wmin .gt. 0.)
c cond4 = (wmax .lt. 0.)
c
c If (cond1 .or. cond2 .or. cond3 .or. cond4) Then
c   centre = .true.
c Else
c   centre = .false.

```



```

        Endif
c
        Goto 99
c
70      Print 220
        Goto 90
c
80      Print 210
90      Goto 1
c
99      Return
100     Format (/, 'Do you want to input from the keyboard or a file? ', $)
110     Format (A1)
118     Format (/, 'Please input header information.')
119     Format (A80)
120     Format (/, 'How many data points do you have? ', $)
130     Format (I2)
140     Format ('Enter the data as x, y')
150     Format ('What file is the data in? ', $)
160     Format (A32)
190     Format (/)
200     Format ('x('I2, '). y('I2, ') -> ', $)
210     Format ('File does not exist; please try again')
220     Format ('Not enough data in the file; try another')
        End

```

```

Subroutine Input1(a,b,n,xmin,xmax,ymin,ymax,nou,header)
c   Input routine. Reads data either from the keyboard or a file.
c
c   Character*1 ans
c   Character*32 fname
c   Character*80 header
c   Real a(nmax), b(nmax)
c
c   Common nmax
c
c   Call Clrscn
c
c   Print 100
c   Read 110, ans
c
c   If (ans .eq. 'k') Then
c
c       Open (Unit = nou, File = 'data.dat', Status = 'unknown')
c
c       Print 118
c       Read 119, header
c
c       Print 120
c       Read 130, n
c       Write (nou,*) n
c
c       Print 140
c
c       Do 10, i = 1, n
c           Print 200, i, i
c           Read *, a(i), b(i)
c           Write (nou,*) a(i), b(i)
10      Continue
c
c       Close (Unit = nou)
c
c       Call Sort(a,n,xmin,xmax)
c       Call Sort(b,n,ymin,ymax)
c
c   Else
c
c       Print 150
c       Read 160, fname
c
c       Open (Unit = nou, File = fname, Status = 'old', Err = 80)
c
c       Read (nou,119, Err = 70) header
c       Read (nou,*, Err = 70) n, xmin, xmax, ymin, ymax
c
c       Do 15, i = 1, n
c           Read (nou,*, Err=70) a(i), b(i)
15      Continue
c
c       Close (Unit = nou)
c

```

```

    If ((num2 .lt. num1) .and. (first .eq. minim)) Then
        stprvx(i) = stprvx(minim)
        stprvy(i) = stprvy(minim)
    Endif
    If (mod(i,3) .eq. 0) Then
        Call plot(stprvx(i),stprvy(i),3)
        Call plot(stcurx(i),stcury(i),2)
    Endif
95  Continue
c
    If (first .eq. minim) Goto 97
c
    first = minim
    second = maxim
c
    Goto 94
c
97  If (centre) Then
        Call Qualgc(tempix,tempy,num)
    Else
        Call Qualg(stcurx,stcury,stprvx,stprvy,1000,minim,maxim)
    Endif
c
    Return
    End

```

```

c
Do 70, j = curr(i), curr(i+1) - 1, stepc
    count1 = count1 + 1
    If (mod(j.num) .eq. 0) Then
        jth = j
    Else
        jth = mod(j.num)
    Endif
    stcurx(count1) = tempx(k-1,jth)
    stcury(count1) = tempy(k-1,jth)
70 Continue
c
If (prev(i) .gt. prev(i+1)) Then
    If (.not.(centre)) prev(i+1) = prev(i)
    stepp = -1
Else
    stepp = 1
Endif
c
Do 80, j = prev(i), prev(i+1) - 1, stepp
    count2 = count2 + 1
    If (mod(j.num) .eq. 0) Then
        jth = j
    Else
        jth = mod(j.num)
    Endif
    stprvx(count2) = tempx(k-2,jth)
    stprvy(count2) = tempy(k-2,jth)
80 Continue
c
num1 = num1 + abs(curr(i+1)-curr(i))
num2 = num2 + abs(prev(i+1)-prev(i))
c
60 Continue
c
If (centre .or. minchk) Goto 65
c
Call Secsrt(stprvy,stprvx,num2)
Call Secsrt(stcury,stcurx,num1)
c
65 Call Setup(xmax,0.,ymax,0.,dev,a,b,csiz)
Call line(stprvx,stprvy,num2,1,0,0,0,0)
Call line(stcurx,stcury,num1,1,0,0,1,0)
c
minim = min(num1,num2)
maxim = max(num1,num2)
c
first = 1
second = minim
c
94 Do 95 i = first, second
    If ((num1 .lt. num2) .and. (first .eq. minim)) Then
        stcurx(i) = stcurx(minim)
        stcury(i) = stcury(minim)
    Endif

```

Endif

c

If (.not.(fixed)) Goto 5

c

52

chkcnt = ncoun-2

Do 55, i = 2, chkcnt

ncheck = (ncoun .ge. 3)

nreduc = ((abs(prev(i+1)-prev(i)) .le. 3) .or.

1 (abs(curr(i+1)-curr(i)) .le. 3))

If (nreduc .and. ncheck) Then

ncoun = ncoun - 1

Do 54, nucnt = i, ncoun

prev(nucnt) = prev(nucnt+1)

curr(nucnt) = curr(nucnt+1)

54

Continue

Goto 52

Endif

55

Continue

c

Do 60, i = 1, ncoun-2

c

If (centre) Then

c

1 Q1 = ((temp(x(k-2,prev(i)-2)-temp(x(k-2,prev(i)) .lt. 0.)
.or. (temp(x(k-2,prev(i))-temp(x(k-2,prev(i)+2) .lt. 0.))

1 Q2 = ((temp(y(k-2,prev(i)-2)-temp(y(k-2,prev(i)) .lt. 0.)
.or. (temp(y(k-2,prev(i))-temp(y(k-2,prev(i)+2) .lt. 0.))

1 Q3 = ((temp(x(k-1,curr(i)-2)-temp(x(k-1,curr(i)) .lt. 0.)
.or. (temp(x(k-1,curr(i))-temp(x(k-1,curr(i)+2) .lt. 0.))

1 Q4 = ((temp(y(k-1,curr(i)-2)-temp(y(k-1,curr(i)) .lt. 0.)
.or. (temp(y(k-1,curr(i))-temp(y(k-1,curr(i)+2) .lt. 0.))

1 Q5 = (prev(i+1)-prev(i) .gt. 50)

Q6 = (curr(i+1)-curr(i) .gt. 40)

c

If (((.not.(Q1) .and. Q2 .and. .not.(Q3) .and. .not.(Q4))

1 .or. (Q1 .and. .not.(Q2) .and. Q3 .and. Q4)) .and. Q5)

2 Then

prev(i) = prev(i) + num

Endif

c

If (((Q1 .and. Q2 .and. Q3 .and. .not.(Q4)) .or.

1 (.not.(Q1) .and. .not.(Q2) .and. .not.(Q3) .and. Q4)

2 .or. (.not.(Q1) .and. Q2 .and. Q3 .and. Q4))

3 .and. Q6) Then

curr(i) = curr(i) + num

Endif

c

Endif

c

If (curr(i) .gt. curr(i+1)) Then

If (.not.(centre)) curr(i+1) = curr(i)

stepc = -1

Else

stepc = 1

Endif

```

Subroutine Imdiff(tempx,tempy,num,xmax,xmin,
1      ymax,ymin,dev,a,b,csiz,centre,minchk)
c      Finds the difference of two images.
c      ** Still under construction **
c
c      Common k, nmax
c
c      Integer curr(1000), prev(1000), ncoun, count1, count2
c      Integer chkcnt, stepp, stepc, nucnt, num, num1, num2
c      Integer dev, minim, maxim, first, second, minpts
c      Logical centre, choice, fixed, prvchk, ncheck, nreduc
c      Logical Q1, Q2, Q3, Q4, Q5, Q6, minchk
c      Real tempy(6,nmax), tempx(6,nmax)
c      Real stcurx(1000), stcury(1000), stprvx(1000), stprvy(1000)
c      Real xmax, xmin, ymax, ymin, a, b, pi, zero, csiz, zerox
c
c      pi = 3.1415926
c      fixed = .true.
c      count1 = 0
c      count2 = 0
c      num1 = 0
c      num2 = 0
c      minpts = 75
c      zero = 1e-3
c
c      If (centre) zero = 3.0e-3
c
c      zerox = 15.0*zero
c      1      ncoun = 0
c
c      Do 50, i = 1, num
c          Do 40, j = 1, num
c
c              If (centre) Then
c                  choice = ((abs(tempy(k-2,i) - tempy(k-1,j)) .lt. zero) .and.
1                  (abs(tempx(k-2,i) - tempx(k-1,j)) .lt. zerox))
c              Else
c                  choice = (abs(tempy(k-2,i) - tempy(k-1,j)) .lt. zero)
c              Endif
c
c              If (choice) Then
c                  curr(ncoun) = j
c                  prev(ncoun) = i
c                  ncoun = ncoun + 1
c                  Goto 50
c              Endif
c
c      40      Continue
c      50      Continue
c
c      If (centre) Then
c          prvchk = (prev(ncoun-1)-prev(1) .lt. minpts)
c          If (prvchk) ncoun = 0
c          Call Zeroer(ncoun,zero,fixed,minpts)
c          If (.not.(fixed)) ncoun = 0

```

Subroutine Waite

c Causes the program to suspend execution until a carriage
c return/line feed is entered.
c

Print 100

Read 110

c

Return

100 Format (/, 'Type <CR> to continue ',\$)

110 Format (A1)

End

```
Subroutine Clrscn
c  Clears the screen on a vt-100 terminal
c
c  Integer escape
c
c  escape = 155
c
c  Print 100, escape, '['2','J',escape,['','H'
c
c  Return
100 Format (1H+, 10A1, $)
End
```



```
Subroutine Conver(a,b,n)
c  Converts an array to a two dimensional indexed array.
c
c  Real a(6,nmax), b(nmax)
c
c  Common k,nmax
c
c  Do 50, i = 1, n
      a(k,i) = b(i)
50  Continue
c
c  Return
c  End
```

APPENDIX A.4

LIST OF PROGRAMS USED IN CHAPTER IV

A Program For Target Tracking For A Case Of Translation

```

program main
real bmat(2),x(50),xp(50),y(50),yp(50),b1(20),b2(20)
real u(50),up(50),v(50),vp(50),rdelt(50),ralfa(50)
real idelt(50),ialfa(50),bs1(20),bs2(20),error1(20),error2(20)
common /ab/ bmat(2)
character yn
logical flag

c
flag=.false.
sum=3.19
del=.28
sum1=1.9
dell=.18
j=1
do 10 i=1,10
b1(j)=sum-del
b2(j)=sum1-dell
sum=b1(j)
sum1=b2(j)
j=j+1
10 continue
sum=0.
del=-.28
sum1=0.
dell=-.18
do 20 i=1,10
b1(j)=sum+del
b2(j)=sum1+dell
sum=b1(j)
sum1=b2(j)
j=j+1
20 continue
io=2
m=8
do 30 i=1,20
bmat(1)=b1(i)
bmat(2)=b2(i)
call image(x,y)
call wplane(u,v,x,y,m)
call pimage(mm,flag,xp,yp,x,y,m)
call wplane(up,vp,xp,yp,m)
call deltat(up,vp,u,v,rdelt,idelt,m)
if(flag.eq..true.)then
qq=-yp(mm)/xp(mm)
idelt(mm)=-atan(qq)

```

```

        flag=.false.
    endif
    call stp(x,y,p,ralfa,ialfa,m)
    call q3q4(q3,q4,ralfa,ialfa,rdelt,idelt,m)
    call aabb(bb1,bb2,p,q3,q4)
    bs1(i)=bb1
    bs2(i)=bb2
    error1(i)=(b1(i)-bs1(i))
    error2(i)=(b2(i)-bs2(i))
30    continue
    print *, ' plot b1'
    read(5,101)yn
101   format(a1)
    call plotb1(b1,bs1)
    print *, ' plot b2'
    read(5,101)yn
    call plotb2(b2,bs2)
    print *, ' plot error for b1'
    read(5,101)yn
    call plter1(b1,error1)
    print *, ' plot error for b2'
    read(5,101)yn
    call plter2(b2,error2)
    stop
end

```

```

subroutine pimage(mm,flag,xp,yp,x,y,m)
real xp(50),yp(50),bmat(2),x(50),y(50)
common /ab/ bmat(2)
    logical flag
c
    do 10 i=1,m
    xp(i)=x(i)+bmat(1)
    yp(i)=y(i)+bmat(2)
    if(x(i).gt.0..and.y(i).eq.0.)then
        if(xp(i).gt.0..and.yp(i).lt.0.)then
            flag=.true.
            mm=i
        endif
    endif
10    continue
    return
end

```

```

subroutine wplane(x1,y1,x2,y2,n)
real x1(50),x2(50),y1(50),y2(50)
c
pi=3.141592654

```

```

do 1 i=1,n
  if(x2(i).ne.0.0.or.y2(i).ne.0.0)
+  x1(i)=alog(sqrt(x2(i)**2+y2(i)**2))
  if(x2(i).eq.0.0.and.y2(i).gt.0.0)y1(i)=pi/2.
  if(x2(i).eq.0.0.and.y2(i).eq.0.0)then
    y1(i)=pi/2.
    x1(i)=-100000.
  endif
  if(x2(i).eq.0.0.and.y2(i).lt.0.0)y1(i)=1.5*pi
  if(x2(i).ne.0.0)then
    q=y2(i)/x2(i)
    if(x2(i).lt.0.0)then
      if(y2(i).gt.0.0)then
        q=-q
        y1(i)=pi-atan(q)
      else
        if(y2(i).eq.0.0)y1(i)=pi
        if(y2(i).lt.0.0)y1(i)=pi+atan(q)
      endif
    else
      if(x2(i).gt.0.0)then
        if(y2(i).gt.0.0)y1(i)=atan(q)
        if(y2(i).eq.0.0)y1(i)=0.0
        if(y2(i).lt.0.0)then
          q=-q
          y1(i)=2.*pi-atan(q)
        endif
      endif
    endif
  endif
  continue
1  return
end

```

```

subroutine deltat(up,vp,u,v,rdelt,idelt,m)
  real up(50),vp(50),u(50),v(50),rdelt(50),idelt(50)
c
do 10 i=1,m
  rdelt(i)=up(i)-u(i)
  idelt(i)=vp(i)-v(i)
10 continue
  return
end

subroutine aabb(b1,b2,p,q3,q4)
c
  b1=q3/p
  b2=q4/p

```

```

return
end

subroutine stp(x,y,p,ralfa,ialfa,m)
real x(50),y(50),ralfa(50),ialfa(50),r(50)
c
    p=0.0
do 10 i=1,m
    r(i)=x(i)**2+y(i)**2
    p=p+1./r(i)
    ralfa(i)=x(i)/r(i)
    ialfa(i)=-y(i)/r(i)
10 continue
    return
end

subroutine plotb1(x,y)
real x(20),y(20)
character yn
c
    print *, ' 4107 ?'
    read(5,100) yn
100 format(a1)
    if(yn.eq.'y') idev=3
    if(yn.eq.'n') idev=2
    if(yn.ne.'y'.and.yn.ne.'n')idev=4
    call plots(idev)
    call scrnp(.25,.7,.25,.75)
    call paper(-4.5,4.5,-5.5,4.5)
    call axis(-3.5,-3.5,19hDELTA X (estimated),19,7.,90.,
+ -3.5,1.5,.1,999,1)
    call axis(-3.5,-3.5,20h    DELTA X (exact),-20,7.,0.,-3.5,
+ 1.5,.1,999,0)
    call plot(-3.5,0.,3)
    call plot(3.5,0.,2)
    call plot(0.,-3.5,3)
    call plot(0.,3.5,2)
    call line(x,y,20,1,0,4,1)
    call line(y,y,20,1,0,4,0)
    call symbol(2.,-2.5,.1,9h___ Exact,1,0.,9)
    call symbol(2.,-3.,.1,12h--- Estimate,1,0.,12)
    print *, ' done ?'
    read(5,100)yn
    call plot(0.,0.,999)
    return
end

subroutine plotb2(x,y)

```

```

real x(20),y(20)
character yn
c
print *,' 4107 ?'
read(5,100) yn
100 format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.25,.7,.25,.75)
call paper(-4.5,4.5,-5.5,4.5)
call axis(-3.5,-3.5,19hDELTA Y (estimated),19,7.,90.,
+-3.5,1.5,.1,999,1)
call axis(-3.5,-3.5,15hDELTA Y (exact),-15,7.,0.,-3.5,
+1.5,.1,999,0)
call plot(-3.5,0.,3)
call plot(3.5,0.,2)
call plot(0.,-3.5,3)
call plot(0.,3.5,2)
call line(x,y,20,1,0,4,1)
call line(y,y,20,1,0,4,0)
call symbol(2.,-2.5,.1,9h___ Exact,1,0.,9)
call symbol(2.,-3.,.1,12h--- Estimate,1,0.,12)
print *,' done ?'
read(5,100)yn
call plot(0.,0.,999)
return
end

subroutine plter1(x,y)
real x(20),y(20)
character yn
c
print *,' 4107 ?'
read(5,100) yn
100 format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.25,.7,.25,.75)
call paper(-4.5,4.5,-5.5,4.5)
call axis(-3.5,-3.5,5hERROR,5,7.,90.,-3.5,1.5,.1,999,1)
call axis(-3.5,-3.5,15hDELTA X (exact),-15,7.,0.,-3.5,
+-1.5,.1,999,0)
call plot(-3.5,0.,3)
call plot(3.5,0.,2)

```

```

call plot(0.,-3.5,3)
call plot(0.,3.5,2)
call line(x,y,20,1,0,4,0)
print *, ' done ?'
read(5,100)yn
call plot(0.,0.,999)
return
end

```

```

subroutine plter2(x,y)
real x(20),y(20)
character yn

```

c

100

```

print *, ' 4107 ?'
read(5,100) yn
format(a1)
if(yn.eq.'y') iddev=3
if(yn.eq.'n') iddev=2
if(yn.ne.'y'.and.yn.ne.'n')iddev=4
call plots(iddev)
call scrnp(.25,.7,.25,.75)
call paper(-4.5,4.5,-5.5,4.5)
call axis(-3.5,-3.5,5hERROR,5,7,90.,-3.5,1.5,.1,999,1)
call axis(-3.5,-3.5,15hDELTA Y (exact),-15,7,0.,-3.5,
+1.5,.1,999,0)
call plot(-3.5,0.,3)
call plot(3.5,0.,2)
call plot(0.,-3.5,3)
call plot(0.,3.5,2)
call line(x,y,20,1,0,4,0)
print *, ' done ?'
read(5,100)yn
call plot(0.,0.,999)
return
end

```

```

subroutine image(x,y,m)
real x(50),y(50)

```

c

```

x(1)=3.
y(1)=2.
x(2)=-3.
y(2)=2.
x(3)=-3.
y(3)=-2.
x(4)=3.
y(4)=-2.
x(5)=3.

```



```
y(5)=0.  
x(6)=0.  
y(6)=2.  
x(7)=3.  
y(7)=0.  
x(8)=0.  
y(8)=2.  
return  
end
```

A Program For Target Tracking For A Case Of Translation,Rotation and Scaling

```

program main
real k,amat(2,2),bmat(2),x(50),xp(50),y(50),yp(50)
real u(50),up(50),v(50),vp(50),rdelt(50),ralfa(50)
real idelt(50),ialfa(50),error3(61)
real error4(61),b3(61),b4(61),b5(61),b6(61),angle(61)
integer mm(3),n1(10),n2(10),count1,count2
common /ab/ amat(2,2),bmat(2)
character yn
logical flag1,flag2,flag3,flag4,flag5

c
flag1=.false.
flag2=.false.
flag3=.false.
flag4=.false.
flag5=.false.
angle(0)=-15.5
del=.5
h=.5
k=.5
m=32
io=2
do 99 ii=1,61
angle(ii)=angle(ii-1)+del
theta=angle(ii)
theta=3.141592654*theta/180.
call amatx(theta,h,k)
call image(x,y)
call wplane(u,v,x,y,m)
call pimage(mm,n1,n2,count1,count2,flag1,
+flag2,flag3,flag4,flag5,xp,yp,x,y,m)
call wplane(up,vp,xp,yp,m)
call deltat(up,vp,u,v,rdelt,idelt,q1,q2,m)
if(flag1.eq..true.)then
qq=-yp(mm(1))/xp(mm(1))
idelt(mm(1))=-atan(qq)
flag1=.false.
endif
if(flag2.eq..true.)then
do 41 i=1,count1
qq=yp(n1(i))/xp(n1(i))
qq2=-y(n1(i))/x(n1(i))
idelt(n1(i))=atan(qq)+atan(qq2)
41 continue
flag2=.false.
endif

```

```

if(flag3.eq..true.)then
  do 42 i=1,count2
    qq=-yp(n2(i))/xp(n2(i))
    qq2=y(n2(i))/x(n2(i))
    idelt(n2(i))=-(atan(qq)+atan(qq2))
42  continue
    flag3=.false.
  endif
  if(flag4.eq..true.)then
    qq=-y(mm(2))/x(mm(2))
    idelt(mm(2))=atan(qq)
    flag4=.false.
  endif
  if(flag5.eq..true.)then
    qq=y(mm(3))/x(mm(3))
    idelt(mm(3))=-atan(qq)
    flag5=.false.
  endif
  call stp(x,y,s,t,p,ralfa,ialfa,m)
  call q3q4(q3,q4,ralfa,ialfa,rdelt,idelt,m)
  call aabb(a1,a2,b1,b2,s,t,p,q1,q2,q3,q4)
  b3(ii)=b1
  b4(ii)=b2
  b5(ii)=bmat(1)
  b6(ii)=bmat(2)
  error3(ii)=bmat(1)-b1
  error4(ii)=bmat(2)-b2
99  continue
  print *, ' plot b1 ?'
  read(5,100)yn
100 format(a1)
  if(yn.eq.'y')call plotb1(angle,b3,b5)
  print *, ' plot b2 ?'
  read(5,100)yn
  if(yn.eq.'y')call plotb2(angle,b4,b6)
  print *, ' plot error for b1 ?'
  read(5,100)yn
  if(yn.eq.'y')call plter1(angle,error3)
  print *, ' plot error for b2 ?'
  read(5,100)yn
  if(yn.eq.'y')call plter2(angle,error4)
  stop
end

```

```

subroutine pimage(mm,n1,n2,count1,count2,flag1,
+flag2,flag3,flag4,flag5,xp,yp,x,y,m)
real xp(50),yp(50),amat(2,2),bmat(2),x(50),y(50)
integer mm(3),n1(10),n2(10),count1,count2

```

```

common /ab/ amat(2,2),bmat(2)
      logical flag1,flag2,flag3,flag4,flag5

```

c

```

      count1=0
      count2=0
      do 10 i=1,m
      xp(i)=amat(1,1)*x(i)+amat(1,2)*y(i)+bmat(1)
      yp(i)=amat(2,1)*x(i)+amat(2,2)*y(i)+bmat(2)
      if(x(i).gt.0..and.y(i).eq.0.)then
        if(xp(i).gt.0..and.yp(i).lt.0.)then
          flag1=.true.
          mm(1)=i
        endif
      endif
      if(x(i).gt.0..and.y(i).lt.0.)then
        if(xp(i).gt.0..and.yp(i).gt.0.)then
          flag2=.true.
          count1=count1+1
          n1(count1)=i
        endif
      endif
      if(x(i).gt.0..and.y(i).lt.0.)then
        if(xp(i).gt.0..and.yp(i).eq.0.)then
          flag4=.true.
          mm(2)=i
        endif
      endif
      if(x(i).gt.0..and.y(i).gt.0.)then
        if(xp(i).gt.0..and.yp(i).lt.0.)then
          flag3=.true.
          count2=count2+1
          n2(count2)=i
        endif
      endif
      if(x(i).gt.0..and.y(i).gt.0.)then
        if(xp(i).gt.0..and.yp(i).eq.0.)then
          flag5=.true.
          mm(3)=i
        endif
      endif
10    continue
      return
      end

```

```

subroutine wplane(x1,y1,x2,y2,n)
real x1(50),x2(50),y1(50),y2(50)

```

c

```

pi=3.141592654

```

```

do 1 i=1,n
  if(x2(i).ne.0.0.or.y2(i).ne.0.0)
+ . x1(i)=alog(sqrt(x2(i)**2+y2(i)**2))
    if(x2(i).eq.0.0.and.y2(i).gt.0.0)y1(i)=pi/2.
    if(x2(i).eq.0.0.and.y2(i).eq.0.0)then
      y1(i)=pi/2.
      x1(i)=-100000.
    endif
    if(x2(i).eq.0.0.and.y2(i).lt.0.0)y1(i)=1.5*pi
    if(x2(i).ne.0.0)then
      q=y2(i)/x2(i)
      if(x2(i).lt.0.0)then
        if(y2(i).gt.0.0)then
          q=-q
          y1(i)=pi-atan(q)
        else
          if(y2(i).eq.0.0)y1(i)=pi
          if(y2(i).lt.0.0)y1(i)=pi+atan(q)
        endif
      else
        if(x2(i).gt.0.0)then
          if(y2(i).gt.0.0)y1(i)=atan(q)
          if(y2(i).eq.0.0)y1(i)=0.0
          if(y2(i).lt.0.0)then
            q=-q
            y1(i)=2.*pi-atan(q)
          endif
        endif
      endif
    endif
  endif
  continue
1 return
end

```

```

subroutine q3q4(q3,q4,ralfa,ialfa,rdelt,idelt,m)
real ralfa(50),ialfa(50),rdelt(50),idelt(50)
c
  q3=0.0
  q4=0.0
  do 10 i=1,m
    q3=q3+rdelt(i)*ralfa(i)+idelt(i)*ialfa(i)
    q4=q4+idelt(i)*ralfa(i)-rdelt(i)*ialfa(i)
10  continue
  return
end

subroutine deltat(up,vp,u,v,rdelt,idelt,q1,q2,m)
real up(50),vp(50),u(50),v(50),rdelt(50),idelt(50)

```

```

c
    q1=0.0
    q2=0.0
    do 10 i=1,m
        rdelt(i)=up(i)-u(i)
        q1=q1+rdelt(i)
        idelt(i)=vp(i)-v(i)
        q2=q2+idelt(i)
10    continue
    return
end

```

```

subroutine aabb(a1,a2,b1,b2,s,t,p,q1,q2,q3,q4)

```

```

c
    del=s**2+t**2-p
    a1=(-q1*p+q3*s+q4*t)/del+1.
    a2=(-q3*t+q4*s-q2*p)/del
    b1=(q1*s-q3-q2*t)/del
    b2=(q1*t-q4+q2*s)/del
    return
end

```

```

subroutine stp(x,y,s,t,p,ralfa,ialfa,m)
real x(50),y(50),ralfa(50),ialfa(50),r(50)

```

```

c
    p=0.0
    s=0.0
    t=0.0
    do 10 i=1,m
        r(i)=x(i)**2+y(i)**2
        p=p+1./r(i)
        ralfa(i)=x(i)/r(i)
        ialfa(i)=-y(i)/r(i)
        s=s+ralfa(i)
        t=t-ialfa(i)
10    continue
    return
end

```

```

subroutine amatx(theta,h,k)
real amat(2,2),bmat(2),k
common /ab/ amat(2,2),bmat(2)

```

```

c
    amat(1,1)=cos(theta)
    amat(2,2)=amat(1,1)
    amat(2,1)=sin(theta)
    amat(1,2)=-amat(2,1)
    bmat(1)=(1.-amat(1,1))*h+amat(2,1)*k

```

```

bmat(2)=amat(1,2)*h+(1.-amat(1,1))*k
return
end

```

```

subroutine plotb1(x,y,z)
real x(61),y(61),z(61)
character yn
c
print *,' 4107 ?'
read(5,100) yn
100 format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.3,.7,.3,.75)
call paper(-22.,19.,-.45,.30)
call axis(-16.,-.25,7hDELTA X,7,.5,90.,-.25,.15,.1,999,1)
call axis(-16.,-.25,14hANGLE (DEGREE),-14,32,.0.,-16.,8.,.1,999,0)
call plot(-16.,0.,3)
call plot(16.,0.,2)
call plot(0.,-.25,3)
call plot(0.,.25,2)
call line(x,y,61,1,0,4,1)
call line(x,z,61,1,0,4,0)
call symbol(6.,.24,.1,9h___ Exact,1,0.,9)
call symbol(6.,.19,.1,12h— Estimate,1,0.,12)
print *,' done ?'
read(5,100)yn
call plot(18.,.18,999)
return
end

```

```

subroutine plotb2(x,y,z)
real x(61),y(61),z(61)
character yn
c
print *,' 4107 ?'
read(5,100) yn
100 format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.3,.7,.3,.75)
call paper(-22.,19.,-.45,.30)
call axis(-16.,-.25,7hDELTA Y,7,.5,90.,-.25,.15,.1,999,1)
call axis(-16.,-.25,14hANGLE (DEGREE),-14,32,.0.,-16.,8.,.1,999,0)

```

```

bmat(2)=amat(1,2)*h+(1.-amat(1,1))*k
return
end

```

```

subroutine plotb1(x,y,z)
real x(61),y(61),z(61)
character yn

```

```

c
print *,' 4107 ?'
read(5,100) yn
100 format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.3,.7,.3,.75)
call paper(-22.,19.,-.45,.30)
call axis(-16.,-.25,7hDELTA X,7,.5,90.,-.25,.15,.1,999,1)
call axis(-16.,-.25,14hANGLE (DEGREE),-14,32.,0.,-16.,8.,.1,999,0)
call plot(-16.,0.,3)
call plot(16.,0.,2)
call plot(0.,-.25,3)
call plot(0.,.25,2)
call line(x,y,61,1,0,4,1)
call line(x,z,61,1,0,4,0)
call symbol(6.,.24,.1,9h___ Exact,1,0.,9)
call symbol(6.,.19,.1,12h— Estimate,1,0.,12)
print *,' done ?'
read(5,100)yn
call plot(18.,.18,999)
return
end

```

```

subroutine plotb2(x,y,z)
real x(61),y(61),z(61)
character yn

```

```

c
print *,' 4107 ?'
read(5,100) yn
100 format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.3,.7,.3,.75)
call paper(-22.,19.,-.45,.30)
call axis(-16.,-.25,7hDELTA Y,7,.5,90.,-.25,.15,.1,999,1)
call axis(-16.,-.25,14hANGLE (DEGREE),-14,32.,0.,-16.,8.,.1,999,0)

```



```

call plot(-16.,0.,3)
call plot(16.,0.,2)
call plot(0.,-.25,3)
call plot(0.,.25,2)
call line(x,y,61,1,0,4,1)
call line(x,z,61,1,0,4,0)
call symbol(6.,.24,.1,9h___ Exact,1,0.,9)
call symbol(6.,.19,.1,12h— Estimate,1,0.,12)
print *, ' done ?'
read(5,100)yn
call plot(18.,.18,999)
return
end

```

```

subroutine plter1(x,y)
real x(61),y(61)
character yn

```

c

100

```

print *, ' 4107 ?'
read(5,100) yn
format(a1)
if(yn.eq.'y') idev=3
if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.3,.7,.3,.75)
call paper(-22.,19.,-.45,.30)
call axis(-16.,-.25,5hERROR,5,.5,90.,-.25,.15,.1,999,1)
call axis(-16.,-.25,14hANGLE (DEGREE),-14,32.,0.,-16.,.8,.1,999,0)
call plot(-16.,0.,3)
call plot(16.,0.,2)
call plot(0.,-.25,3)
call plot(0.,.25,2)
call line(x,y,61,1,0,4,0)
print *, ' done ?'
read(5,100)yn
call plot(18.,.18,999)
return
end

```

```

subroutine plter2(x,y)
real x(61),y(61)
character yn

```

c

100

```

print *, ' 4107 ?'
read(5,100) yn
format(a1)
if(yn.eq.'y') idev=3

```

```

if(yn.eq.'n') idev=2
if(yn.ne.'y'.and.yn.ne.'n')idev=4
call plots(idev)
call scrnp(.3,.7,.3,.75)
call paper(-22.,19.,-.45,.30)
call axis(-16.,-.25,5hERROR,5,.5,90.,-.25,.15,.1,999,1)
call axis(-16.,-.25,14hANGLE (DEGREE),-14,32,.0,-16.,8.,.1,999,0)
call plot(-16.,0.,3)
call plot(16.,0.,2)
call plot(0.,-.25,3)
call plot(0.,.25,2)
call line(x,y,61,1,0,4,0)
print *,' done ?'
read(5,100)yn
call plot(18.,.18,999)
return
end

```

```

subroutine image(x,y)
real x(50),y(50)

```

c

```

x(1)=3.
y(1)=2.
x(2)=-3.
y(2)=2.
x(3)=-3.
y(3)=-2.
x(4)=3.
y(4)=-2.
  x(5)=3.
y(5)=0.
x(6)=0.
y(6)=2.
x(7)=-3.
y(7)=0.
x(8)=0.
y(8)=-2.
  y(9)=1.
  x(9)=3.
  x(10)=-3.
  y(10)=1.
  x(11)=3.
  y(11)=-1.
  x(12)=-3.
  y(12)=-1.
  x(13)=1.5
  y(13)=2.
  x(14)=-1.5

```

```
y(14)=2.  
x(15)=-1.5  
y(15)=-2.  
x(16)=1.5  
y(16)=-2.  
x(17)=-3.  
y(17)=.5  
x(18)=-3.  
y(18)=-.5  
x(19)=3.  
y(19)=-.5  
x(20)=3.  
y(20)=.5  
x(21)=-.75  
y(21)=2.  
x(22)=-.75  
y(22)=2.  
x(23)=-.75  
y(23)=-2.  
x(24)=.75  
y(24)=-2.  
x(25)=2.25  
y(25)=2.  
x(26)=-2.25  
y(26)=2.  
x(27)=-2.25  
y(27)=-2.  
x(28)=2.25  
y(28)=-2.  
x(29)=3.  
y(29)=1.5  
x(30)=-3.  
y(30)=1.5  
x(31)=-3.  
y(31)=-1.5  
x(32)=3.  
y(32)=-1.5  
return  
end
```

APPENDIX B

DETAILED COMPUTER GRAPHICS OF THE
THREE STRUCTURE DESIGNS

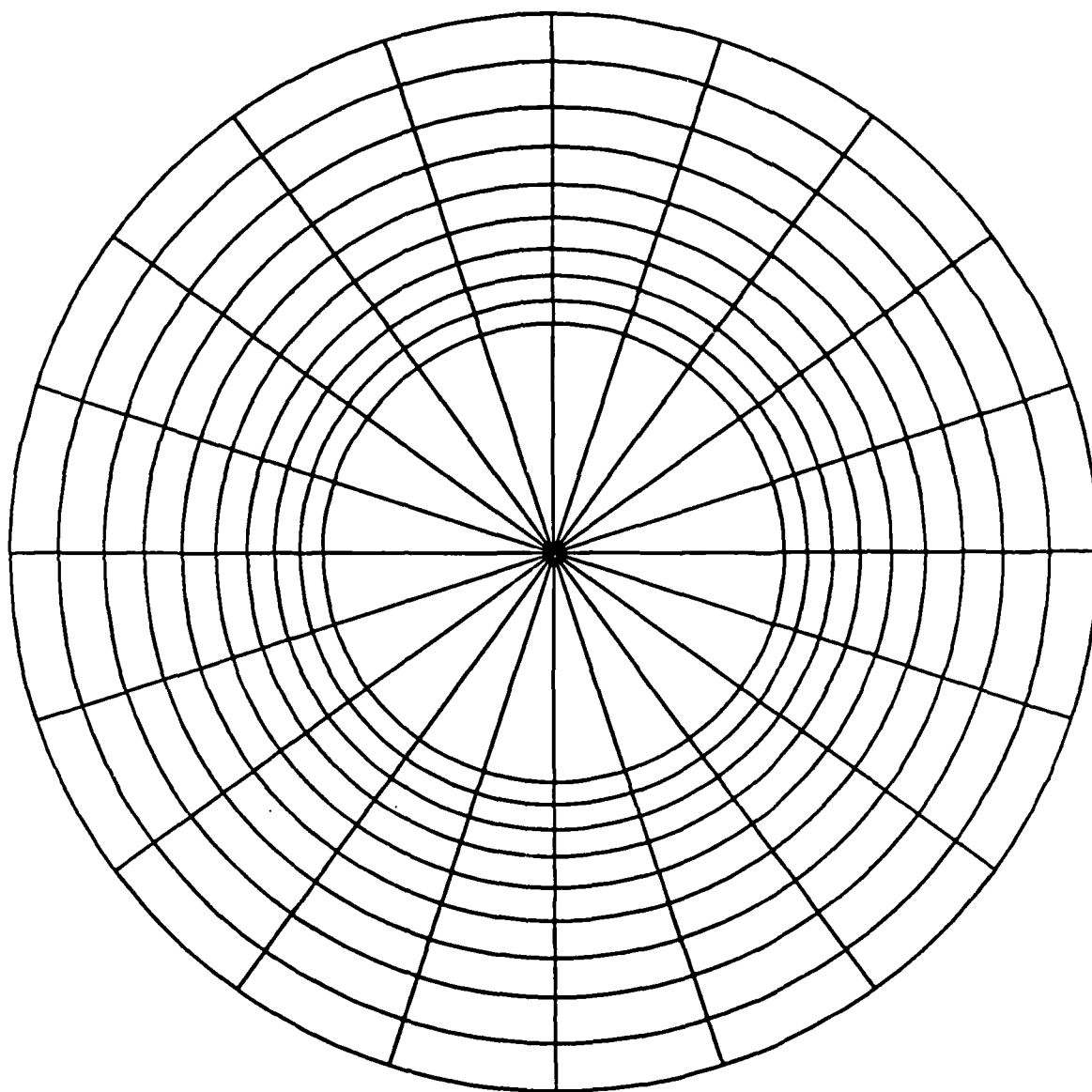


Figure B1. Structural Design Featuring Rectangular Elements.
Image Plane.

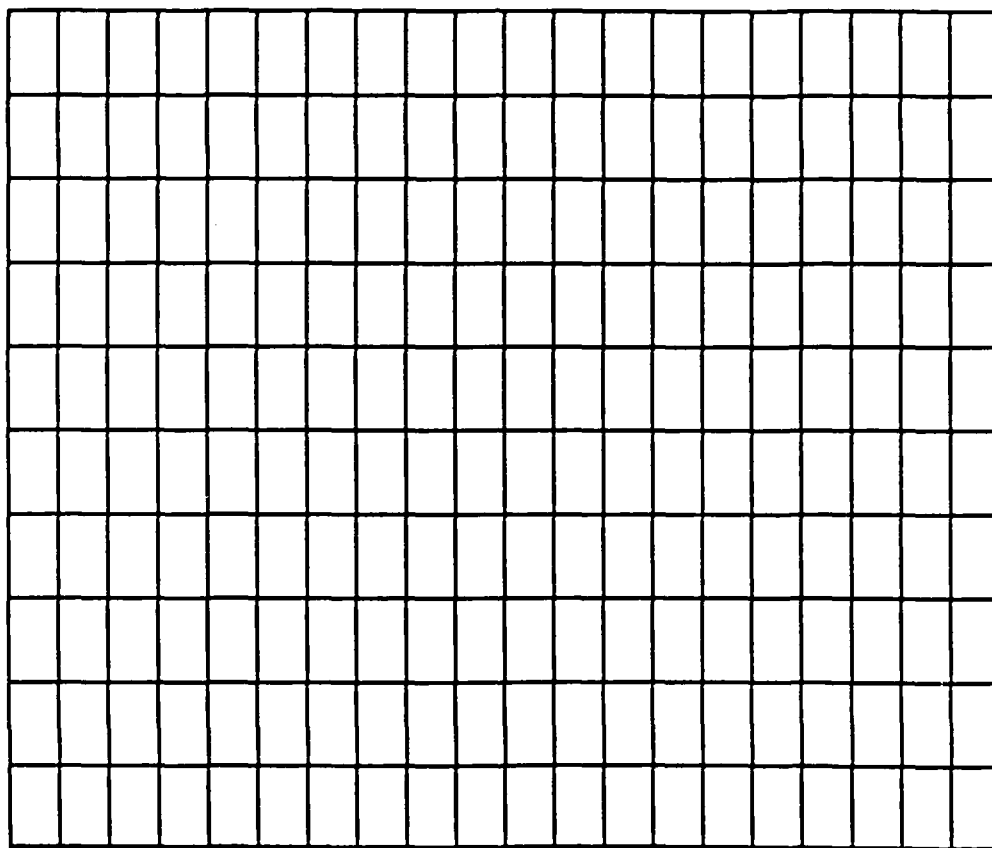


Figure B2. Structural Design Featuring Rectangular Elements.
Computational Plane.

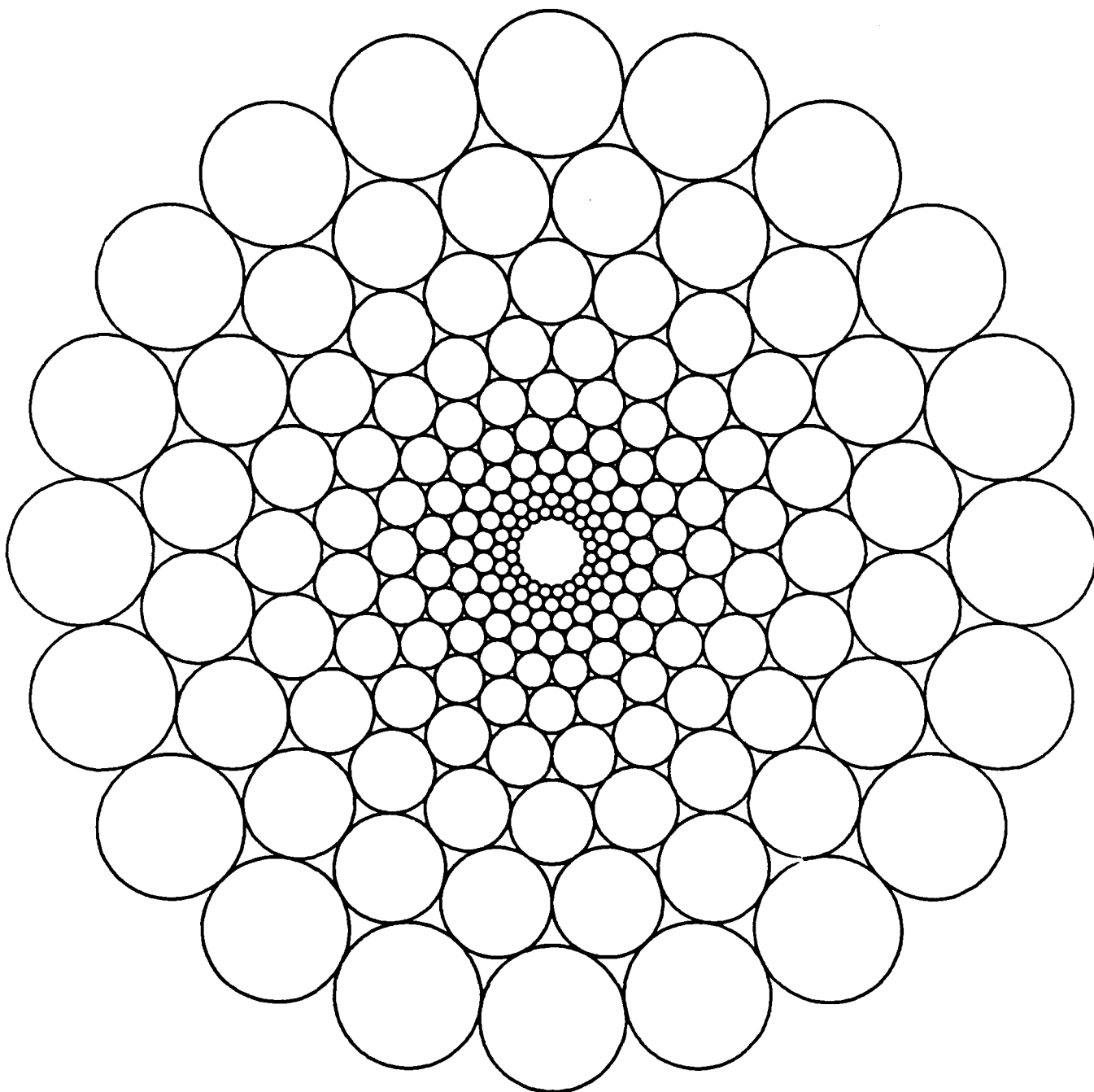


Figure B3. Structural Design Featuring Circular Elements.
Image Plane.

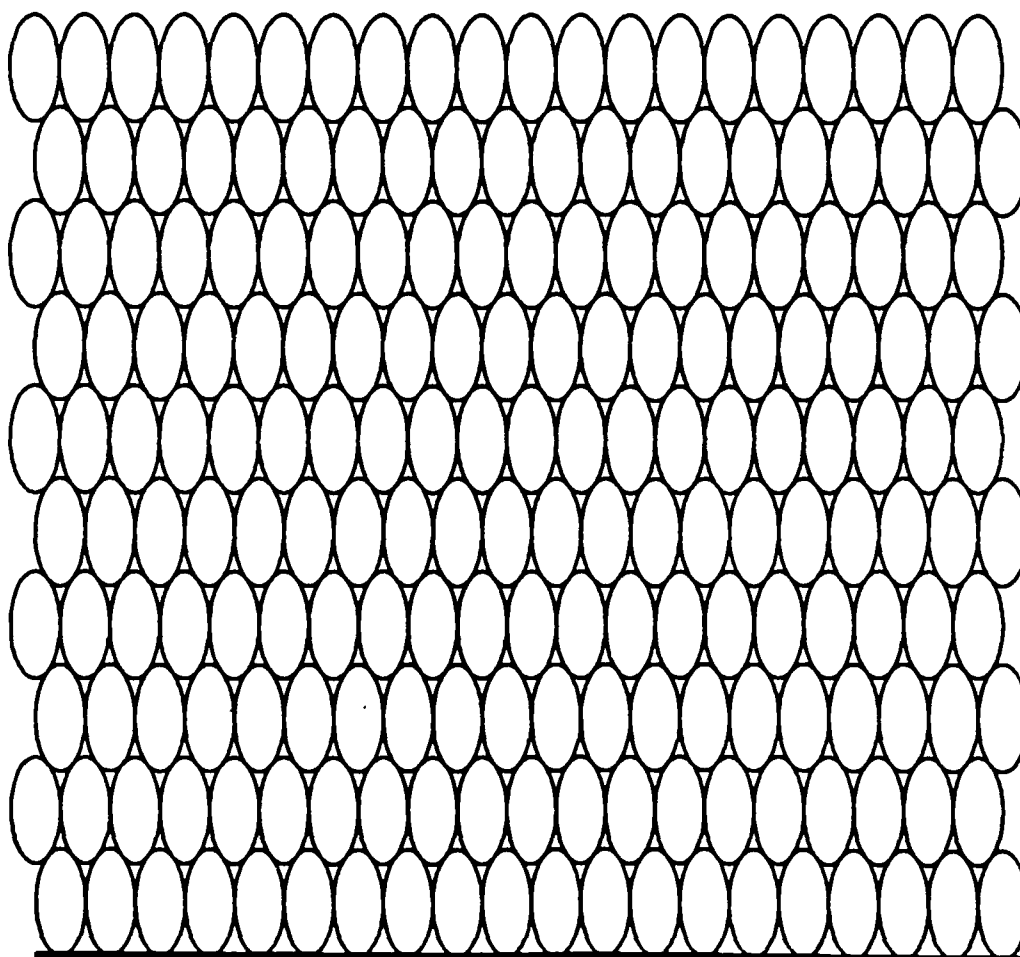
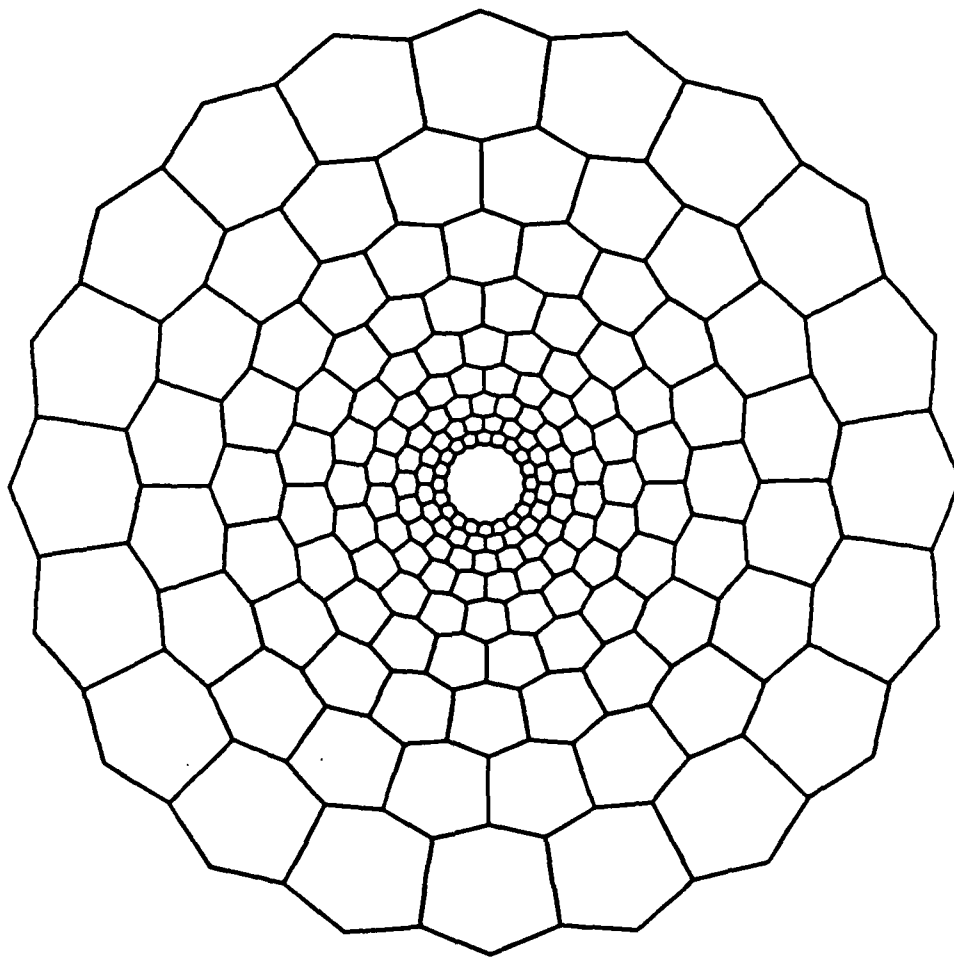


Figure B4. Structural Design Featuring Circular Elements.
Computational Plane.



**Figure B5. Structural Design Featuring Hexagonal Elements.
Image Plane.**

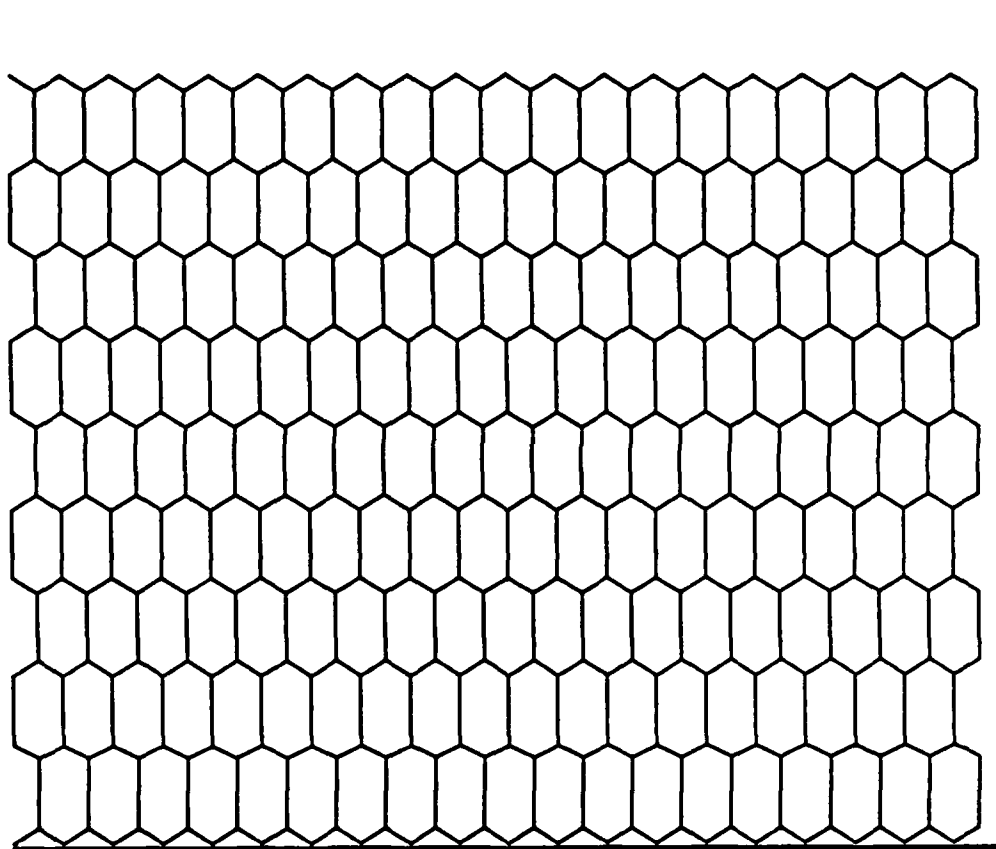


Figure B5. Structural Design Featuring Hexagonal Elements.
Computational Plane.

DISTRIBUTION LIST

Copy No.

1 - 16	Air Force Office of Scientific Research Bolling Air Force Base Washington D. C. 20332 Attention: Lt. Col. Robert W. Carter
17 - 20	Mr. Rick Wehling AFATL-DLMI Eglin AFB, FL 32542
21 - 22	R. M. Inigo, EE
23 - 24	E. S. McVey, EE
25	J. F. Doner, BME
26	Chen Ho Hsin, GRA, EE
27	Jay I. Minnix, GRA, EE
28	Chiewdarn Narathong, GRA, EE
29	Valerie Davis, GRA, EE
30 - 31	E. H. Pancake Clark Hall
32	SEAS Publications Files

JO#7185/rsr

UNIVERSITY OF VIRGINIA

School of Engineering and Applied Science

The University of Virginia's School of Engineering and Applied Science has an undergraduate enrollment of approximately 1,400 students with a graduate enrollment of approximately 600. There are 125 faculty members, a majority of whom conduct research in addition to teaching.

Research is an integral part of the educational program and interests parallel academic specialties. These range from the classical engineering departments of Chemical, Civil, Electrical, and Mechanical and Aerospace to departments of Biomedical Engineering, Engineering Science and Systems, Materials Science, Nuclear Engineering and Engineering Physics, and Applied Mathematics and Computer Science. In addition to these departments, there are interdepartmental groups in the areas of Automatic Controls and Applied Mechanics. All departments offer the doctorate; the Biomedical and Materials Science Departments grant only graduate degrees.

The School of Engineering and Applied Science is an integral part of the University (approximately 1,530 full-time faculty with a total enrollment of about 16,000 full-time students), which also has professional schools of Architecture, Law, Medicine, Commerce, Business Administration, and Education. In addition, the College of Arts and Sciences houses departments of Mathematics, Physics, Chemistry and others relevant to the engineering research program. This University community provides opportunities for interdisciplinary work in pursuit of the basic goals of education, research, and public service.

END

DTIC

8-86